NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

SYSTEMATIC ASSESSMENT OF THE IMPACT OF USER ROLES ON NETWORK FLOW PATTERNS

by

Jeffrey S. Dean

September 2017

Dissertation Supervisor: Neil Rowe

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2017	Dissertation 09-21-2009 to 09-22-2017

4. TITLE AND SUBTITLE

SYSTEMATIC ASSESSMENT OF THE IMPACT OF USER ROLES ON NETWORK FLOW PATTERNS

5. FUNDING NUMBERS

6. AUTHOR(S)

Jeffrey S. Dean

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: NPS.2012.0005-AM01-EP7-A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Defining normal computer user behavior is critical to detecting potentially malicious activity. To facilitate this, some anomaly-detection systems group the profiles of users expected to behave similarly, setting thresholds of normal behavior for each group. One way to group users is to use organizational role labels, as people with similar roles in an organization often share common tasks and activities. Another way is to group users based on observed behavioral similarities. We tested the premise that users sharing roles behave similarly on networks, applying two machine-learning classifiers (nearest-centroid and a support vector machine) to differentiate between groups based on flow-data feature vectors. We conducted tests using 1.2 billion network-flow records from a large building at Naval Postgraduate School over five weeks. Tests showed similar results when they were conducted with and without removal of automated flows. Tests showed that users in role groups do not exhibit significantly similar network behaviors. We also clustered feature-vector data to group users by patterns of network behavior and showed that defining user groups this way provides a better way to bound normal user behavior.

14. SUBJECT TERMS

netflow, user behavior, machine learning, organizational role

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

SYSTEMATIC ASSESSMENT OF THE IMPACT OF USER ROLES ON NETWORK FLOW PATTERNS

Jeffrey S. Dean
Civilian, United States Air Force
M.S., Air Force Institute of Technology, 2004

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE
from the
NAVAL POSTGRADUATE SCHOOL
September 2017

Approved by: Neil Rowe
Professor of Computer Science
Dissertation Supervisor

James Michael
Professor of Computer Science

John McEachen
Professor of Electrical and Computer Engineering

Marcus Stefanou
Assistant Professor of Computer Science

Robert Kaufman
Adjunct Professor
Information Systems and Cyber Security

Approved by: Peter Denning
Chair, Department of Computer Science

Approved by: Douglas Moses
Vice Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Defining normal computer user behavior is critical to detecting potentially malicious activity. To facilitate this, some anomaly-detection systems group the profiles of users expected to behave similarly, setting thresholds of normal behavior for each group. One way to group users is to use organizational role labels, as people with similar roles in an organization often share common tasks and activities. Another way is to group users based on observed behavioral similarities. We tested the premise that users sharing roles behave similarly on networks, applying two machine-learning classifiers (nearest-centroid and a support vector machine) to differentiate between groups based on flow-data feature vectors. We conducted tests using 1.2 billion network-flow records from a large building at Naval Postgraduate School over five weeks. Tests showed similar results when they were conducted with and without removal of automated flows. Tests showed that users in role groups do not exhibit significantly similar network behaviors. We also clustered feature-vector data to group users by patterns of network behavior and showed that defining user groups this way provides a better way to bound normal user behavior.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ACK	TCP Acknowledge Flag
BBC	British Broadcasting Company
BCC	Bi-Connected Components
CERT	Computer Emergency Readiness Team
CNN	Cable News Network
CTMC	Continuous-Time Markov Chain
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DL	Distance Learning
DNS	Domain Name Servers
DPI	Deep Packet Inspection
DTW	Dynamic Time Warping
FAMS	Fast Adaptive Mean Shift
FIN	TCP Finish Flag
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IBM	International Business Machines
ICMP	Internet Control Message Protocol

ID	Identity
IDF	Inverse Document Frequency
IETF	Internet Engineering Task Force
IDES	Intrusion Detection Expert System
IGMP	Internet Group Management Protocol
I/O	Input/Output
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information eXport
IRIS	Identity Risk and Investigation Solution
IT	Information Technology
ITACS	Information Technology and Communications Services
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MMP	Markov Modulated Process
MS	Microsoft
NBA	Network Behavior Analysis
NetBIOS	Network Basic Input/Output System
NIST	National Institute of Standards and Technology
NPS	Naval Postgraduate School
NTP	Network Time Protocol
PCA	Principle Component Analysis

PhD	Doctor of Philosophy
PPK	Probability Product Kernel
PPV	Port Priority Vector
RBAC	Role-Based Access Control
RPC	Remote Procedure Call
RST	TCP Reset Flag
SiLK	System for Internet-Level Knowledge
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSH	Secure Shell
SYN	TCP Sync Flag
SVDD	Support Vector Data Description
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TTL	Time to Live
TF-IDF	Term Frequency - Inverse Document Frequency
UDP	User Datagram Protocol
VPN	Virtual Private Network
WEKA	Waikato Environment for Knowledge Analysis

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Among the threats network-security analysts face in maintaining the confidentiality, integrity and availability of their networks, the damage that can be inflicted by the users inside the network can be the most grave. Insiders have direct access to much of the critical information available on an organization's network, and through ignorance, carelessness or malicious action they can cause data to be lost, corrupted or destroyed. Detecting the actions of users that are behaving badly has been an area of much research, efforts that are occasionally reinforced by major breaches of trust that make the headlines.

Network monitors that perform anomaly detection operate by comparing the network traffic patterns of users or systems against some model of normal behavior. Set the detection thresholds too tightly, however, and the system will generate more false positive detections. Set the thresholds too broadly, and the system will miss legitimate behavioral anomalies. One method used to define normal-behavioral thresholds is to group users based on expected similarities in behavior. Defined this way, acceptable ranges of measured behavior would be broader than ranges based on individual user activities, but not as broad as ranges that could be exhibited by a group of randomly selected users.

One approach to grouping users with expected similarities in behavior is to do so based on user roles in their jobs. In some commercial network-monitoring systems that use network-flow metadata (e.g. Netflow) to detect network threats, network users can be grouped based on shared common organizational roles. Grouping of user profiles based on roles has an intuitive appeal based on the assumption that shared roles imply shared tasks and activities, but little has been published showing a link between user roles and patterns of network traffic documented by Netflow.

This dissertation describes the methods we used to test the relationship of user roles to the network-traffic patterns they produce, as captured using Netflow data. In the course of our investigation we developed an approach to identifying flow records created by automatic system processes, to enable removing those records from data sets intended to capture user behaviors on the network. We identified and tested different Netflow-derived-feature sets developed as a means of describing user-network-traffic behaviors. We tested the effect of

using different sampling intervals for aggregating Netflow records to create feature vectors, measuring the impact of employing these intervals on machine-learning classifiers.

We conducted experiments with 1.2 billion flow records obtained by our information-technology department monitoring software for a large building at our school over five weeks. We trained and tested two machine-learning classifiers (nearest-centroid and a support vector machine) on feature-vector-data sets derived from groups defined by shared roles, and from a control group of users randomly selected. We found that the performance of the classifiers was similar for both role-based and control groups, indicating that role-group-data sets shared no more common behaviors than were shared by groups of randomly selected users. We clustered feature vectors using K-Means++ and found that most clusters contained mixtures of data from each role-based-user group, with only a few small clusters dominated by data from one or two role groups. We performed pairwise distance comparisons between feature-vector sets from each of our identified users, and found that on average the data sets of users within the same role group were as different from each other as they were to data sets of users from other role groups. From these observations, we have concluded that for comparing user behaviors based on Netflow data, the grouping of users based on shared organizational roles has limited usefulness except for dealing with new users with little data.

We then grouped users by using K-Means++ to cluster centroid vectors of the feature vectors for each user. Tests showed that the classifiers performed better in differentiating between these user-group-data sets and control sets. Distance comparisons between users within the same user group were lower than comparisons between data sets of users from different user groups. We conclude that for the identification of normal behavior of users on a network, the groups used to define normal behavior ranges should be based on measured behavioral similarities and not by user roles.

Acknowledgments

To Marla, who supported me throughout this process. Despite the pain, boredom, and loneliness while watching me try to reach my goals, you hung in there and made it possible. Thank you.

To my dissertation adviser, Dr. Rowe, whose patience and professionalism made me believe I could do this even when I was losing faith. You taught me how to find answers when none were appearing, and always to keep making progress. Thank you for pushing me through this process. To the rest of my committee, thank you for your patience with all my last-minute changes in content and schedule as I worked toward the finish line. I could not have made it without all your help.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Motivation

Being a manager of an enterprise network has always been challenging, but the challenges today appear to be increasing greatly in both scope and complexity. Organizations are being hacked by criminals, hacktivists (hackers with some activist agenda), and nation-states. Insider threats, where employees with legitimate access compromise the confidentiality, integrity or availability of critical organizational data and resources, can and often do cause great damage to many government and private organizations. As many organizations allow employees to bring their own devices to work, the variety in available applications, operating systems and computing platforms complicates defining normal network traffic [1]. Intrusion-detection sensors searching network traffic for malicious activity must deal with increasing network speeds, driving more demanding performance requirements (processing rates, network and disk I/O bandwidth, disk-storage capacity for captured traffic and derived artifacts). The increasing use of encryption is rendering many signature-based detection methods less useful. Finally, many organizations are limited in the extent to which they can analyze network traffic due to privacy laws limiting what network traffic artifacts can be captured and stored.

Among these challenges the identification of potentially harmful behaviors among digital system users has been an area of intense interest, as organizations struggle with detecting personnel abusing access privileges through malicious actions, ignorance or carelessness. To address this concern, much research has gone into creating tools to detect anomalous-user behaviors that could indicate harmful activities. Many of these tools create user profiles based on activities observed on host systems and networks, and use these profiles to compare user activities against historical behaviors or against the behaviors of other users in the organization.

Network-monitoring systems that employ anomaly detection algorithms compare new network-traffic patterns against some standard of "normal" traffic [2]. Defining normal

traffic based on the daily activities of individual users can be problematic, as there is no guarantee that the user was not behaving anomalously when their profile was created. In addition, even non-malicious changes in behavior could be flagged as an anomaly [3]. To get around this problem, some systems enable grouping user profiles, using the aggregate behaviors of the group to define behavioral thresholds. If a selected user group shares similar network behaviors, the ranges of behavioral measures for that group should be broader than any individual in the group but narrower than ranges defined by users with dissimilar behavior patterns [3]. For some commercial tools, users may be grouped based on sharing the same office or organizational roles, based on the assumption that people responsible for the same general set of tasks should behave more similarly than those performing different tasks in the organization [4].

This assumption is plausible since one would expect that an administrative assistant would exhibit detectable behaviors on a computer or network quite different from those of a network administrator. But does grouping and evaluating users based on roles provide the best approach for defining normal digital behaviors? People are individuals, with different interests, skills and habits. These individual differences should also impact the observable network artifacts generated by user activity, possibly to the extent of obscuring any similarities due to users sharing the same defined roles.

Another approach to identifying normal behavior ranges by creating user groups is to measure similarities between user profiles, and group users based on these similarities [3]. This approach has the benefit of being objectively verifiable, with user similarities measured based on the same behavioral features that are being used to characterize and compare user activities on the network. We evaluate the assumption that users sharing similar roles exhibit similar network behaviors, and contrast the level of similarity found in role-based user groups against the similarities of users grouped based on shared measures of network behavior.

1.2 Contributions

The primary contributions of our research are:

- We identified and tested a more detailed set of feature vectors for discriminating user

network behavior than any tried previously, and showed their power. We showed that composite feature vectors are more useful than individual features.

- We demonstrated that the flow-derived feature sets associated with each user were on average significantly more self-similar than they were similar to feature sets of other users in their own role group. Thus role groups provide limited predictive power except in cases of new users for which data was inadequate but the role group is known.
- We demonstrated that for defining the scope of normal user network activity as measured by network flow metadata, users in groups identified based on similar behavior profiles are measurably more similar in behavior than groups defined based on shared user roles.
- We developed a methodology for testing whether the criteria used for identifying user groups bears any significant relationship to the behavioral characteristics of that user group. In this case, we demonstrated that defining user groups based on organizational roles did not result in creating groups exhibiting shared network behavior patterns.
- To support the use of Netflow data in comparing user (and not host) network behaviors, we developed and tested new methods for eliminating automated flows. We also tested the impact on removing those automatic flows on our processes for comparing user behaviors.

1.3 Document Structure

The prior work supporting the concepts and approaches used in our research is provided in Chapter 2. We present our definition of user behavior as observed in the cyber domain, how anomalous user behavior may be defined and the implicit relationship between organizational roles and user behaviors. The features of Netflow data are described, along with patterns observed in network traffic as described by Netflow records. The primary characteristics of flows generated automatically by applications and operating systems are described. Finally, we describe the known research relevant to the problem of comparing user behaviors using Netflow records.

Chapter 3 describes the patterns used to identify and remove automatically generated flow records in the data set, and how the algorithms used to detect automatic flows were developed and tested. It describes the feature sets used for comparing of user behaviors, the user-role

groups used to test the relationship between roles and behaviors, and the data set collected for evaluating this relationship. It discusses the user-behavior-similarity measures used for measuring the differences between extracted Netflow-based features.

In Chapter 4, the experiments used to test similarities and differences between user behaviors are discussed. Comparisons between data sets reflecting the network patterns generated by different user role-groups are tested, including control data sets constructed to be non-role specific. Each test is repeated with and without automatic flows removed from the data, to determine the impact of removing their data. In addition, different data sampling intervals are tested to determine which interval best enables differentiating traffic from different role groups. Chapter 5 provides the results of the experiments discussed in Chapter 4, and analyzes them, and Chapter 6 discusses the conclusions we derived from our experiments.

CHAPTER 2:
Prior Work

2.1 Observing Cyber Behaviors on Networked Systems

What is meant when we talk about behavior in the cyber domain? For our research we define behavior as patterns within digital traces such as log files, file transfers observed in captured packet data, command line input sequences, and network traffic metadata, generated by the activities of digital-system users, platform and applications used to access the cyber domain. Cyber behaviors can be observed and compared using multiple data sources, including host systems, network-monitoring devices, and network appliances (e.g. routers, firewalls, databases, web servers) [5], [6]. This rich set of options provides network defenders multiple opportunities for detecting threats, with many commercial and open source tools available for collecting, analyzing and presenting data extracted from these resources.

Intrusion-detection systems employ algorithms that look for patterns within digital traces that could indicate network system compromises; these algorithms can be categorized as being based on signatures, stateful-protocol analysis or anomaly detection [7]. Well-defined digital patterns based on specific byte-value sequences in transferred data are characteristic of signature-based systems, and are good for detecting known threats [8]. The success of signature-based threat detection depends on the specificity of the signatures used. Matches to highly specific signatures provide high confidence in a valid detection, but such specific signatures may not detect closely related threats. Unknown threats are not detected by signature-based systems, as signatures are derived from what is known. Stateful-protocol analysis compares network traffic against a representation of normal behavior, where definitions of normal behavior are based on protocol standards [7]. This approach is also known as Specification-Based Detection, and is often used in hybrid systems in combination with signature or anomaly-based detection techniques.

Anomaly-based intrusion detection algorithms compare extracted digital patterns against some representation of normal behavior [9]; deviations from normal behavior greater than

some threshold can be declared anomalous, and flagged for investigation [8]. Lazarevic et al. categorized cyber behavior detection approaches primarily as statistical-based, knowledge-based and machine-learning based [10]. Statistical-based detection compares one or more feature metrics extracted from current user or system activity against a profile, and alerts if the metrics deviate too far from the profile norms. While these systems can be trained to profile normal behavior they suffer from several drawbacks, including attackers or insiders being able to shift the statistics of normal traffic over time, an assumption that the features have known (usually normal) statistical distributions and that they are generated by relatively stable processes. Knowledge-based detection relies on rules derived from human expertise, and can be developed using some kind of formal tool to describe expected behaviors. Garcia-Teodoro et al. [9] noted that while machine-learning is similar to statistical-based detection in terms or comparing data sets, machine-learning based techniques categorize or classify patterns and several algorithms can improve classification accuracy by training to reduce past errors. This ability to categorize patterns is a very useful capability for characterizing how related groups of data sets, or the systems and users that generated the data sets, are to each other.

By using normal behavior as a yardstick, anomaly detection can detect certain kinds of new threats [2]. As Gates and Taylor observed [11] not all anomalies are malicious, and depending on the features used for analysis not all malicious traffic appears anomalous [12]. Used by themselves anomaly-detection systems are susceptible to high false-positive rates, often due to evaluating new but legitimate behaviors. This tendency can be mitigated by broadening the criteria used to define normal behavior, either by increasing the deviation thresholds required for an alert, or increasing the number of data samples used to define normal behavior (collecting over longer periods or drawing data from more users) [2], [3].

2.2 Detecting Anomalous User Network Behaviors

Identifying anomalous digital behavior by users on a network has some similarities to detecting malware, a well established and robust area of research. Some potentially malicious behaviors can be well defined, such as a user trying to access network resources for which they do not have privileges [8]. These can be codified as signatures on network or host-based sensors, to generate alerts when they are detected. The significance of other user behaviors, such as an analyst downloading a larger volume of documents during the day than expected,

may be better determined relative to some representation of normal activity. Such behaviors can be detected using statistical or machine-learning based anomaly-detection systems [8].

The challenge in using these approaches is determining how to set good thresholds on behavior-related measures (features), to separate values associated with normal behaviors from those associated with anomalous activity [13]. Selecting the best thresholds would result in maximizing the number of correctly identified anomalies while minimizing false alarms [8]. Setting behavioral thresholds based on the network-usage profiles of individual users can be problematic, in that 1) A user's behavior may already be anomalous when a profile is being created [8] and 2) A user data set may show limited variability in observed feature values, so that minor changes in the user's behavior triggers a false alarm [3]. Defining normal behaviors based on the consolidated data sets of users with different patterns of behavior (such as an entire organization) can also be problematic, in that feature value ranges within the group may be wide enough that feature values associated with anomalous behaviors are not detected.

One way of addressing these problems is to define normal behaviors based on the consolidated data sets of users that might be expected to behave similarly. In financial fraud detection, this approach is referred to as peer group analysis [14]. In user misuse detection, users fulfilling the same roles within an organization are sometimes grouped together. If users within the same role group do behave similarly on the network, the ranges of feature values used to describe their collective network behaviors could be expected to be narrower than would be found by grouping unrelated sets of users, yet broader than the feature value ranges of individual users within the group [3].

2.2.1 User Roles and Network Behavior

User roles have an implicit relationship to what should be considered normal behavior, as different roles have different associated tasks to be performed and different access permissions to perform those tasks [15]. Thus the network traffic generated by users from different role-based groups can be hypothesized to differ in detectable ways, based on the network resources accessed, the protocols used and levels of activity associated with that access. Based on this, grouping users within an organization to define normal traffic patterns for each role group makes sense.

The relationships between users, roles and tasks however are many to many; a user may have multiple roles in an organization, and a role may have many users assigned to it. Likewise a role may be tied to multiple tasks and the associated privileges needed to accomplish each task, while tasks and associated privileges can be common to multiple roles [15]. For example, the tasks for a computer user fulfilling the role of student could include collecting information (accessing library resources and internet browsing), processing information (writing, performing experiments, and coding), and communicating information (handling mail and making presentations). In an educational institution these tasks are by no means limited to students, as faculty, administration and staff personnel may often find themselves needing to fulfill the same or similar tasks. Because of the overlap in tasks between different role groups and the fact that users are individuals with their own approaches to completing those tasks, defining normal behaviors based on user roles is not as straightforward as one might assume.

A number of researchers have looked into the application of user roles or peer groups as a means of refining the parameters of normal user behavior. Park and Giordano [16] leveraged roles in experiments designed to detect insider threat behaviors. Frequency measurements were taken for activities such as document searches, and the ranges of values observed for each role group were used to set thresholds for normal behavior. In the experiment, if a user exceeded these thresholds their behavior would then be compared to that individual's normal range, as a means of reducing false positive alerts. They concluded that the use of roles did help in defining normal behaviors. The data used in this study however was synthetic, generated to emulate an Intelligence Community (IC) organization, and so did not necessarily reflect patterns that would be observed in a real organization.

Nellikar et al. [17] also tested whether the use of role-based data labeling enhanced detection of behavioral anomalies. Log files representing document access times were generated using a continuous-time Markov chain based (CTMC) algorithm, providing sequences of randomly timed events. Different CTMC structures were configured for the two roles (mechanical engineer and chemical engineer), and separate traces were created for each virtual user. Malicious behavior logs were those with timing characteristics different from the two role based groups. Each group was supposed to have access to different document sets, plus one common document that both groups had full access to. The testing was performed on accesses to the common document.

Using the synthetic log data tracking document access frequencies, they tested several classifiers:

- One-Class Support Vector Machines,
- Support Vector Data Description (SVDD),
- Fast Adaptive Mean Shift (FAMS),
- One-Class Classifier (a WEKA [18] classifier), and
- Outlier detection using interquartile ranges (WEKA InterquartileRange attribute filter)

The classifiers were trained on the user access patterns, and tested using normal and malicious log sets. The authors found that the classifiers identified the malicious access patterns between 20% and 40% of the time if the classifiers were trained on logs from both role based groups. When the training data was restricted to data from a single role based group, users acting maliciously within the group were detected 100% of the time. Some classifiers (the one-class classifier and the Interquartile classifier) showed higher false positive rates using the single role training sets. This was attributed to the decrease in overall feature variability in the smaller data sets; more data points were identified as outliers and flagged.

While the source data was synthetic, the tests did imply that subgrouping profile data based on roles improved detection accuracy. The underlying assumption that document access time rates are driven by user roles was not proven, however.

IBM's Identity Risk and Investigation Solution (IRIS) identifies anomalous behavior by applying the concept of peer behavior to determine normal activities [4]. Available features are varied, including measures such as application access frequency, time between accesses, number of sensitive data items accessed, login times and remote access events. IRIS treats these features as random variables, creating value distributions based on the activities of peer groups (i.e. users with similar roles that are expected to exhibit similar behavior patterns). Measurements from individual user behaviors are compared to these distributions, to determine if the behaviors are anomalous.

Mathew et al. [19] examined the results of SQL database queries, and created statistical summaries of the query results, or S- vectors, for classification. The statistical vectors were

used to train different (Naive Bayes, Decision Tree Classifier, Support Vector Machines, and Euclidean k-means clustering) classifiers, with the S-vector training data labeled based on the role (Staff, Faculty, Chair, Framework) of the person making the query. Framework queries were queries common to all users. The classifiers were trained as binary discriminators, i.e. Chair versus Faculty, Chair versus Staff, and Faculty vs. Staff. The k-means classifier performed best in identifying the roles of the database users, achieving a 91% - 100% detection rate. Accessing database query results was performed via an application monitoring the interactions between a web-page interface and the database.

2.2.2 User Versus Role-Group Behaviors

Tuning detection thresholds, to maximize true-positive detections and minimize false-positive alerts generated by an anomaly detection system, is one of the challenges of anomaly detection [20]. Ideally, the data used to represent of what is normal for an anomaly detection system corresponds well with the range of expected legitimate behaviors. IBM's Incident Response and Intelligence Services (IRIS) system allows the grouping of peers or users who share organizational roles, in order to define normal ranges of behavior for that group [4].

While organizational roles provide one approach to grouping users to better define normal network behavior, doing so is based on the assumption that users performing similar roles behave similarly. This top-down approach, where group labels are imposed on the data sets, is dependent on that assumption being valid. The validity of this assumption is complicated by a number of factors, however. Role titles may range from broad-based groups (e.g. professors) to more specific designations (e.g. associate professor in the Computer Science department). Tasks within a role group may not be distributed evenly, or may vary between departments. Many organizations allow employees to bring their own digital devices in to connect to the network, each with different operating systems and applications. Computers often make network connections autonomously, performing background actions such as checking for updates or searching for network resources, and these connection patterns can vary between differently configured systems. Users may connect to the organization networks via different media (e.g. wired, wireless, and virtual private network), which can impact temporal aspects of the connections such as how long data transfers take and the lengths of time that user devices stay connected. These sources of variability can impact

the level of apparent similarity of user behaviors within the same role-based group, and the utility of subdividing users into these groups. The relative impacts of these factors on observed connection patterns are largely unexplored.

Another approach to defining normal behaviors by grouping data sets is to not assume users in role-groups behave similarly, but to group user data sets that correspond to similar aggregate behaviors. Such an approach has been used in detecting fraud in financial systems for some time. Ferdousi et al. [21] aggregated features relating to the behaviors of stock brokers during set periods of time, and compared sequences of measurements showing broker activities over time. From these sequences, they created peer groups based on some set number of peers with the closest set of sequence vectors. Frias-Martinez [3] grouped users based on similarities in flow-derived-feature sets. Such groupings could allow tighter definitions of normal behavior than would be attained via grouping data sets by user roles, yet still provide broader feature-value ranges of normal user behavior than might be observed in an individual user profile. Using this approach adds a step in consolidating data sets based on similar behaviors, but is more objectively verifiable than consolidation based on user labels.

2.2.3 Netflow

Our research used a form of network-flow metadata, specifically Netflow version 5 flow records, to compare user network behaviors. Netflow v5 provides a succinct summation of the flow of network data between two computer systems, providing metadata about the flow including source IP address, destination IP address, source port, destination port, flow protocol, number of packets, number of bytes, TCP flags used, flow start time, duration and end time. The Netflow v5 standard also includes a number of other features relating to flow routing, features visible if the flow data is captured by a router. Instead, we used packet capture (pcap) file data to generate the Netflow records, and the SiLK [22] software suite.

Why compare user behaviors using flow metadata? Several key factors favor its use. Netflow is a fairly ubiquitous resource, and creating flow-metadata based sensors is inexpensive. Most current routers can produce Netflow records (or some related variant), plus there are a number of software packages that enable computers to convert packets into flow metadata. The records themselves are compact, typically about 1% of the data-storage volume required

by full packet capture. This property enables organizations to store historical traffic data for longer periods of time, a very useful capability when network breaches are discovered well after the event. Use of Netflow records also allows network traffic analysis without violating many of the privacy laws in place to protect the personal information of employees and customers. Finally, more comprehensive solutions may not be available. Host-based user monitoring, depending on the implementation, may be infeasible for many organizations due to privacy issues and/or cost and complexity. In addition, as encryption becomes more prevalent due to security concerns, reading application layer data is becoming more problematic. Netflow data does not report packet contents.

Netflow provides a record of connection patterns; which systems were contacted, how much data was transferred, what protocols and ports were used, any TCP flags set during the transfer, when the flow started and ended.

2.3 Technical Approaches

We will define the elements of a user's role in an organization as the "rules and norms that comprise a blueprint or script that guides behavior and choices" [23], implicitly or explicitly specifying "appropriate goals, tasks to be executed, and the like" [24]. Research on incorporating a user's role in an organization in the detection of malicious digital users on a network has primarily been limited to applying Role Based Access Control (RBAC) principles, where a user's access to network resources is granted or denied by their roles. In an RBAC framework, a user that attempts to bypass RBAC constraints can be detected and flagged for investigation. The detection of suspicious behaviors that occurs within the limits of a user's access privileges, such as downloading large numbers of sensitive documents within a short period of time for the purposes of exfiltration, is a different problem not solved by applying basic RBAC principles. Some enhanced forms of RBAC attempt to address this limitation, however.

The use of roles as a means of enhancing the detection of anomalous user behaviors to date has been limited, and primarily dependent on the use of data resources such as system logs, user queries, and host processes beyond just network traffic metadata like Netflow records. An alternative to employing roles as a means of defining normal user behaviors is to group user-data sets based on exhibiting similar behaviors, and measuring new user data against

those of users known to be similar.

We now discuss prior work that contributed to development of the methodologies to be described in Chapter 3.

2.3.1 Role Based Access Control

While most of this document uses Netflow-derived data for the detection of malicious user activities, it is important to incorporate a description of Role Based Access Control (RBAC) concepts. Much of the terminology and conceptual framework associated with malicious user mitigation map directly to RBAC principles and implementations. The principles of RBAC have been iteratively defined and refined since 1987 [15], emerging as a unified concept in 1992 [25], and has become a best practices framework for managing the access permissions of users to network resources.

At its essence, RBAC assigns roles to users, and permissions for resource access are based on those roles. More formally, within an organization there exists a set of users, $U = \{u_1, u_2, ..., u_n\}$, as well as a set of roles $R = \{r_1, r_2, ..., r_m\}$ required to fulfill the organization's mission. Because different roles require access to different resources, a set of permissions ($P = \left\{p_1, p, ..., p_q\right\}$) is defined to constrain what resources can be accessed under each role. The organization maps users to roles and this can be a many-to-many mapping, i.e. a user may be assigned to multiple roles, and multiple users may be assigned to a role. Likewise, roles and permissions typically have a many-to-many mapping relationship, in that a role may be associated with multiple permissions, and permissions for access to a network resource may also be assigned to multiple roles.

Because of these many-to-many relationships, RBAC employs the concept of sessions, where a user can take on a set role (and associated permissions) for performing tasks. To perform other roles, the user must end the current session and start another. Besides the constraints imposed by permissions, RBAC can impose other constraints such as disallowing the same user to have certain combinations of roles, such as the making and approving of loans.

Through the use of RBAC principles, an organization can map a complex set of permissions to new employees (or employees moving between divisions) based on the roles associated

with their positions. Restrictions based on need-to-know and separation of duties can be implemented and managed. In many implementations roles and associated permissions can be defined in a hierarchical fashion, such that more senior positions can inherit permissions associated with lower grade personnel in the role hierarchy.

NIST [15] defined their RBAC model as composed of four components, Core RBAC (concepts of users, roles, sessions, objects, operations and permissions), Hierarchical RBAC (inheritance relationships), Static Separation of Duty Relations (based on role set memberships), and Dynamic Separation of Duty Relations (based on roles assumed during sessions). When used properly, RBAC has been shown to be an effective tool for managing and protecting network-based resources.

While RBAC has proven to be quite useful, it has limitations. Park and Ho [26] noted that RBAC ignores insider behaviors such as "communication patterns, frequencies, areas of topics, areas of interest, etc." It also falls short with one of the biggest insider threat risks; users with extensive privileges such as system administrators. Their knowledge and privileges can be applied to create extensive damage while enabling them to cover their tracks. Regarding these behavioral aspects of insiders, basic RBAC provides some preventive capabilities but limited detection capability.

Some of these detection deficiencies can be addressed by adding additional constraints, which can be based on temporal [27], [28], history [29], and event [30] criteria. Baracaldo & Joshi [31] proposed incorporating the concepts of trust and risk within the RBAC framework. In their solution, the system keeps track of the roles a user employs, and can make a decision to deny access to resources if cumulative accesses would enable the inference of sensitive information. Coupled with the risk of inferring information is the concept of trust, based on a trust threshold assigned to each user. The user can be denied access to a set of roles if they do not have a sufficient level of trust for the context of the access. Context is associated with how the interaction may occur, such as through a remote access.

2.3.2 Behavior Based Access Control

RBAC provides a rigorous means of constraining user behaviors on a network based on their roles and privileges, but it does not necessarily prevent users from abusing the privileges associated with their roles. To address anomalous behaviors performed within the limits of

a user's privileges, some means of defining normal behavior is required.

Frias-Martinez [3] described a behavior-based access-control system which clustered user data sets, grouping them based on patterns of similar behavior. New feature vectors were compared against existing clusters, and vectors too dissimilar to existing clusters were determined to be anomalous. For one experiment, 300 users were profiled over two weeks based on port 80 and port 22 usage. The feature vectors used for profiling users consisted of arrays of values created hourly and daily for traffic over each port, based on total number of flows, average flow size in bytes, average flow time in milliseconds, total number of packets, average number of packets per flow, total number of unique IP addresses connected to, and average packet size in bytes passed.

User profiles consisted of the average values for each feature captured during the collection period, effectively a centroid of the feature-vector values observed. The user profiles were clustered using a K-means++ algorithm. The value of k was varied to find values that maximized the correct rejection of anomalous profiles and minimize false rejections. The system correctly rejected 95% of the anomalous feature vectors (created by artificially changing feature values by one standard deviation), indicating that the approach could detect differences in behavior.

In another experiment, host traffic on a wireless network was clustered based on port 21 (FTP), port 22 (SSH), port 25 (SMTP), and port 80 (HTTP) distributions. Feature vectors were created based on the standard deviation and mean values for several features: the number of unique users a host connected to, the number of packets exchanged and length of packets. The profiles were generated per hour for each port and each direction of flow. Thus for a profile p_i for user i, a set of hourly histograms h_{f_n} for each feature f_n were created. For each hour j of the day, the histogram data was abstracted as mean (a_j) and standard deviation (σ_j) values, and a histogram set (h_{f_n}) was defined as:

$$h_{f_n} = \{(a_0, \sigma_0), (a_1, \sigma_1), ..., (a_{23}, \sigma_{23})\}$$

and user profiles were sets of feature histograms:

$$p_i = \left\{h_{f_1}, h_{f_2}, ..., h_{f_n}\right\}$$

Averages of the mean (a_j) and standard deviation (σ_j) values for each feature were computed for each day, and thresholds for single user profiles were based on whether measured feature values for an hour were more than one standard deviation from the day's mean value. Thresholds for profiles that were clustered were set by the maximum distance between a behavior profile ($a_j + \sigma_j$) and all the other $n - 1$ profiles within its cluster c, or

$$t_{P_i} = max_{j=0..n}(d(P_i, P_j))$$

The number of hosts falsely classified as anomalous was lower for cluster-based profile comparisons than for single user based profile comparisons, which is to be expected from using the maximum threshold value for the cluster rather than basing it on individual variations. This concept was refined in [32], which described a mechanism for automating the clustering process. In addition, an incremental learning mechanism was introduced to automatically update the behavior based access control policies.

The system proposed by Frias-Martinez described a methodology for grouping users with demonstrated commonalities in their behaviors, as a means of identifying normal behaviors for a subset of the users being monitored.

2.4 Netflow Based Profiling Techniques

Although research in detecting user misbehaviors has taken many forms, much of the work on the technical side of network misuse detection has drawn from anomaly-detection techniques. As Denning [33] noted, the key to detecting computer misuse is establishing a pattern of the normal (profiles), and using these profiles as yardsticks to identify anomalous behavior. Behaviors in Denning's model were characterized by vectors of measurements taken over a period, or statistical models used to determine if new observations were anomalous. Profiles characterized the behaviors of one or more subjects (those that initiate actions) in relation to objects (resources), creating signatures of normal activity. Activities on the system were captured in audit records, and detected deviations captured in anomaly records. Responses would be specified in a set of activity rules, and triggered by anomalous event detections. The principles of the Intrusion Detection Expert System (IDES) as presented in [33] have been used in many intrusion detection solutions developed since then

which have focused on a wide range of features.

2.4.1 Creating Profiles

To measure user or host behaviors over time, some method of segmenting the collected data into smaller data subsets is required. Each subset would contain user or host-related data generated during a period of time, to be compared against other data subsets generated by that user or host (to detect changes in behavior) or against subsets generated by other users or hosts (to detect differences in behavior).

Clarke et al. [34] approached this problem by dividing user flow data sets based on the services visited or utilized (BBC, Dropbox, Facebook, Google, Hotmail, Skype, Twitter, Wikipedia and YouTube). For each access, they recorded the start and end times of the interaction, the local and server port numbers, the server IP address, the total packets sent for each flow direction, and the total bytes passed. By breaking up the flow data sets based on the known IP addresses associated with each service, they were able to isolate each per service interaction for each user. The captured features were used to train a multi-layer neural net to identify the users associated with these interactions. The true positive rates achieved in differentiating between 46 users ranged between 12.6% and 86%, depending on the user evaluated.

Vinupaul et al. [35] segmented user-flow-data sets by applying a sliding window of N flow records, extracting features each time the window incrementally shifted by one flow record. For each flow sample, they noted the number of unique IP addresses visited, the IP address associated with the most flows, the number of connections to the most visited IP address, the number of unique destination ports accessed, the top destination port accessed, the median flow duration in the sample, the median local port, number of unique local ports, the number of unique Time To Live (TTL) values observed in the sample packets and the top TTL value observed. They trained Random Forest and C4.5 decision trees plus a K Nearest Neighbor classifier on the features to determine how well they could differentiate vectors attributed to 65 users. The F1-scores for the classifiers ranged between 0.8 and 0.87.

Giroire et al. [36] divided user data sets based on the connections users made via corporate laptops. Connection types were categorized as those internal to the company network, connections to the company network via VPN, and connections made outside of the com-

pany. The use of network access categories provided an easily detectable division point for separating user data sets for analysis. Using this approach to grouping user data sets, they observed that users employ laptops differently in different environments, based on the use of server ports, flow protocols and connection durations when connected to internal, external or VPN networks. From this they concluded that any form of behavioral thresholding process used to identify anomalous behaviors would need to take into account the environment in which a system was used.

Interval Based Statistical Profiling: A simpler means of segmenting user or host data sets would be to divide the collection period based on regular intervals. For each interval period, flows that overlap with that period are grouped together. For flow metadata, flows that overlap with more than one interval can be broken into smaller flows, each flow receiving quantity data (bytes, packets) proportional to the fraction of the flow overlapping each interval. Due to its simplicity, interval-based analysis of flow-derived metrics is widely used. Using this approach, features based on flow values that occur within an interval can be generated (mean and standard deviation measures of byte values, histograms of IP addresses visited, etc.). Feature sets can also be generated based on measures observed across multiple intervals, such as histograms of feature values measured for each interval during a day.

Kind et al. [37] explored the use of interval-based histograms of Netflow features to detect network attacks. The features used to create the histograms were the source IP addresses, destination IP addresses, source port numbers, destination port numbers, protocol numbers, number of bytes, number of packets, and flow durations observed during an interval. They reduced the number of feature values in the histograms using Principal Component Analysis (PCA), and tested different distance functions (Manhattan, Euclidean, Mahalanobis) by applying each measure while performing hierarchical clustering on the transformed histogram vectors. The authors noted that while the Mahalanobis distance measure typically provides better results as compared to standard Euclidean distance for clustering, applying PCA to the data set to reduce the data dimensionality provides the same effect.

They compared the clusters generated by attack free traffic sets to those generated from data including various attacks, and were able to detect 86.7% of the attacks. They also compared their histogram based approach for detecting anomalies to using entropy as a measure of traffic changes, and found that use of histograms was more effective for detecting attacks that

impacted only a few histogram values. This approach demonstrated the utility of comparing histograms of Netflow feature values as a means of clustering different network behaviors.

McHugh et al. [38] employed histograms of tuple values (protocol, destination port, and number of bytes) representing features in network flows observed over daily intervals. Byte values were categorized based on ranges, i.e. 1-99, 100-999, 1000-9999, 10000-49999, and >50000 bytes. The tuple feature ranges were quantized to allow 23 different possible combinations, and the normalized counts for each tuple were used to train a three layer back-propagation neural network. Each output node in the neural network represented one of three hosts on the network to be identified. The neural network correctly classified 100% of the hosts in the test data. The authors intended to use the network to search for host anomalies based on misclassifications. In one case however, a misclassification was caused by an individual who changed which host they worked on partway through the day. This indicates that the neural net was profiling the host and user together, and the relative values among the output neurons could provide similarity measures between different users.

Melnikov et al. [39] explored the feasibility of profiling and identifying specific users based on Netflow statistical features. Their sample size was limited (4-5 volunteers), but with this group they explored several Netflow features in terms of user classification. They found that the relationships of HTTP flow bytes to durations and flow bytes to packets did not aid classifiers. Conversely, they observed notable differences between users in histograms (feature-value histograms based on data collected over defined intervals) showing the distributions of SSH flow-duration times. To investigate HTTPS flow durations as a distinguishing characteristic they cross-correlated duration histograms, and found much higher correlations between histograms from the same users as compared to histograms derived from different users. This implies that applying distance measures to flow-feature distributions may be useful.

Sequence Based Profiling: Classifications based on statistical feature values are usually temporally agnostic, in that relationships between values as a function of time are ignored. Temporal patterns, however, could be characteristic of a user or group of users in the sequences of tasks or activities performed during the day. It is unlikely that users or groups would create identical sequences or patterns, as people can perform tasks faster or slower, or at different times of the day. This means that in comparing two sequences, A and B,

temporal comparisons should be either statistical in nature (e.g. based on a Markov process), or allow for matching similar sub-sequences within two sequence data sets being compared.

Coull et al. [40] proposed an approach to calculate an overall distance value between two flow sequences, that incorporated the element of patterns over time. This was achieved by aligning similar subsequences through a modified version of Dynamic Time Warping, to find a minimum inter-sequence distance value. Similarity between subsequences was determined by use of a distance metric, measuring the difference between features (flow port values, IP addresses, byte counts and time of day) of pairs of flows. The challenge in determining an inter-flow distance value was that some flow features are numerical (bytes, packets, duration, flow time of day), and some categorical (IP addresses, ports, protocol, flags).

Categorical flow features had to be mapped into relationships that aligned with their intended purposes. For IP address and port values, this was achieved by creating hierarchies of addresses and ports (Figure 2.1). Distance values were assigned based on the level in the hierarchy at which two port or address values diverged.

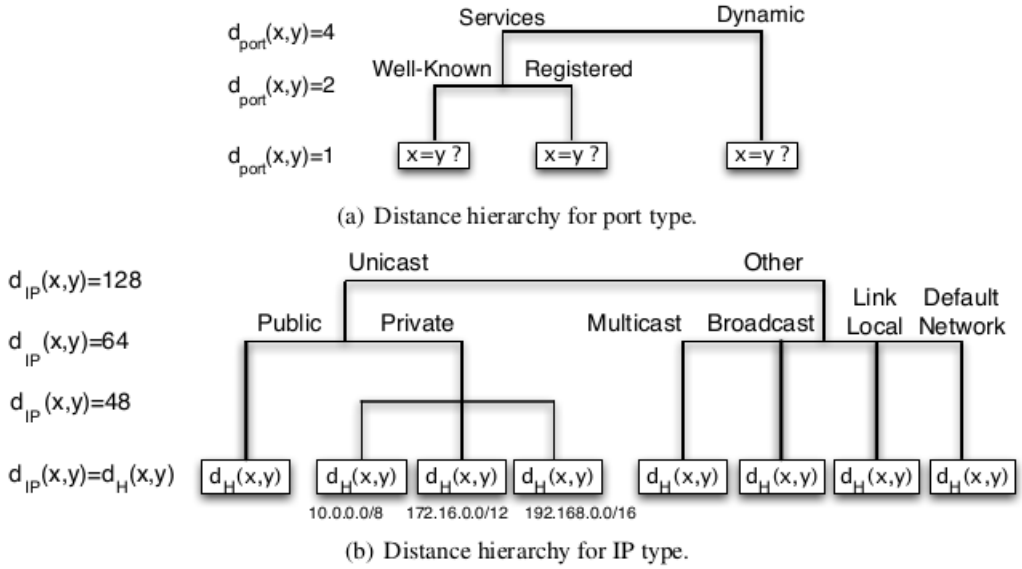


Figure 2.1. IP Address and Port Hierarchies. Source: [40]

For comparing two destination ports for example, in [40] they defined the distance $d_{port}(a, b)$ as:

$$d_{port}(a,b) = \begin{cases} 0 & if\ a = b \\ 1 & if\ \delta_{port}(a) = d_{port}(b) \\ 2 & if\ \delta_{port}(a) \in \{0,1\}\ \&\ \delta_{port}(b) \in \{0,1\} \\ 4 & if\ \delta_{port}(a) \in \{0,1\}\ \&\ \delta_{port}(b) \in \{2\} \end{cases} \qquad \delta_{port}(a) = \begin{cases} 0 & if\ a \in [0,1023] \\ 1 & if\ a \in [1024,49151] \\ 2 & if\ a \in [49152,65535] \end{cases}$$

Numerical values, such as flow bytes or time of day (in seconds), were placed in non-overlapping categories based on their common labels. Flow byte values were measured in bytes, kilobytes or megabytes, while time of day values (since midnight) were measured in seconds, minutes and hours. Coull et al. [40] mapped the categorical values into a common [0, 1] range, by subdividing the range to match the number of categories (Figure 2.2).

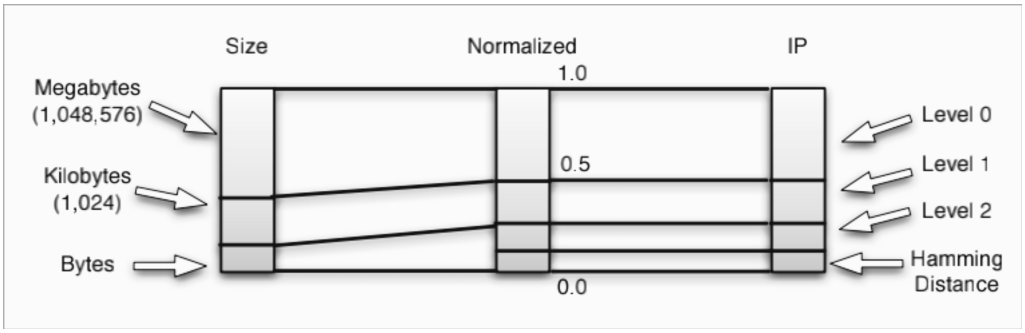


Figure 2.2. Piece-wise range mapping. Source: [40]

For each of the flow features $f_k \in$ {flow time of day, source IP, source port, destination IP, destination port, flow bytes}, the absolute difference $|f_k^{a_i} - f_k^{b_j}|$ between two mapped values was used to quantify the feature distance. The overall flow pair distance $d(a_i, b_j)$, where $a_i \in A$ and $b_j \in B$, was the Euclidean distance of the feature distances, or $d(a_i, b_j) = \sqrt{\sum_{k=1}^{n}(f_k^{a_i} - f_k^{b_j})^2}$.

The temporal element, where the sequences of flows from two different hosts are compared, came through use of a modified dynamic time warping (DTW) algorithm. The modification to DTW was an effort to reduce the $O(m_1, m_2)$ complexity of comparing sequences $A = \{a_1, a_2, ..., a_{m_1}\}$ and $B = \{b_1, b_2, ..., b_{m_2}\}$. The basic DTW algorithm computes all possible distance values, or $d(a_i, b_j)$, between terms in sequence A and sequence B. Recursively, the distance between subsequences $A = \{a_1, a_2, ..., a_{m_1}\}$ and $B = \{b_1, b_2, ..., b_{m_2}\}$ can be

expressed as:

$$D(i,\ j) = d(a_i, b_j) + min(D(i-1, j-1), D(i-1, j), D(i, j-1))$$

for all $1 \leq i \leq m_1$ and $1 \leq j \leq m_2$. $D(-1,\ j)$, $D(i,\ -1)$ and $D(-1,\ -1)$ are assumed to be infinite in value.

One way to envision this is to create an A x B matrix, where cell (i, j) represents a pairing of the i^{th}term in the A sequence against the j^{th} term in the B sequence. $D(i, j)$ is the shortest warped path distance to that cell, and is computed based on $d(a_i, b_j)$ and the minimum previously computed distance values in cells to the left, upper left and above cell (i, j). The total distance between the two sequences is $D(m_1, m_2)$.

In the modified DTW algorithm discussed in [40], for Netflow record sequences A and B (|A|≥|B| is assumed) the sequences are split up into subsequences, in which flows within a subsequence occurred during the same second. Subsequences in A are mapped to subsequences in B that occurred at the same time, i.e. for subsequences $A_1, A_2, ..., A_k$and $B_1, B_2, ..., B_k$, A_i is mapped to B_i. Unpaired subsequences, where no corresponding flow subsequences exist, are merged with the closest (temporally) subsequence for that system.

This mapping continues into the subsequence level. A slope term, $S_i = \frac{|B_i|}{|A_i|}$, enables a pairing of each flow within subsequence A_i to a flow within B_i, such that the j^{th}point in A_i(or $A_{i,j}$), is mapped to $B_{i,j'}$, where $j' = \lceil j * S_i \rceil$. The purpose of this mapping approach is to identify a set of "diagonal" flow pairs through the A x B matrix, allowing for the fact that one system may produce more flows in a given second than the other. The authors argue that optimal mappings between the A and B sequences typically occur near the same points in time, and so identifying flow pairs that occur at or near the same time enables creating a smaller "window" of cells to process. The window size w_i for each subsequence is dynamic, at least $\lceil S_i \rceil$ in size, such that distances within a window of flow pairs $(d(A_{i,j-w_i}, B_{i,j'-w_i}), d(A_{i,j-w_i+1}, B_{i,j'-w_i+1}), ..., d(A_{i,j}, B_{i,j'})..., d(A_{i,j+w_i}, B_{i,j'+w_i}))$ are processed. The overall warped distance, $D(m_1, m_2)$, is normalized to $D'(m_1, m_2) = D(m_1, m_2)/(|A|, |B|)$, so that $0 \leq D'(m_1, m_2) \leq 1$.

Relationships between different hosts on the network were evaluated in [40] by agglomerative clustering based on the sequence distance metric. Using this approach, they were able

to separate server-like behaviors from client systems, and show closer distances between similar behaving hosts. For enterprise-scale networks generating billions of flow records, comparing users based on flow record sequences would require considerable processing resources, especially if comparing each user against all other users on the network.

As an alternative, the modified dynamic time warping algorithm could be used to compare sequences of data vectors, where each data vector contains features derived from aggregates of flows rather than individual flow records. Given a distance measurement between different data vectors, the similarity between sequences of data vectors can be determined. We tested the sequence comparison methods outlined by Coul et al. [40] on a subset of our collected Netflow records, and on a form of flow-data-feature vector we named Port Priority Vectors (described in Section 3.2.2). For both data types, we compared sequences reflecting the network-flow activities of 30 users assigned to five role groups. We compared the sequence-distance measurements between users within each role group, and between users from different role groups, but did not find any significant differences between the intra-group and inter-group comparisons.

Paschalidis and Smaragdakis [41] modeled flow byte counts over defined intervals using a Markov Modulated Process (MMP). Measured values were quantized into set ranges, and the probabilities of transitions between successive values were measured for a known anomaly-free trace. The state transition probability matrix extracted from the anomaly-free traffic is used as a reference for monitored traffic. This approach flags an anomaly if the probability that the state transitions observed in monitored traffic remained low. While this approach is geared towards detecting anomalies, it could be used to compare flow data patterns between different users and hosts.

Song et al. [42] also modeled flow behavior using a Markov model; transition tables were built based on sequences of distant port values. The states in the table represented the lowest 1024 ports, with all ports above 1024 consolidated into one state. The Markov models for different systems were compared using a probability product kernel (PPK) function, and pairwise distance functions between hosts were clustered using spectral clustering. Using this approach, it was possible to identify hosts with similar temporal behaviors. Although this technique was applied using port values, other statistical metrics could be used for building the model and comparing hosts and users.

Feature Set Based Profiling: Discriminating between users or role groups based on flow metadata can also performed using endpoint-address-based features. Kumpost and Matyas [43] created matrices of source IP versus destination IP addresses, one each for SSH, HTTP and HTTPS flows, examining patterns observed during days, weeks or months. Each cell(i,j) in the matrix represented the number of connections from source IP i to destination IP j. Row connection-count values were then normalized to sum up to one, to create a "behavior vector" for that source IP address. Systems A and B could be compared by computing the cosine similarity of the vectors, $cos_{(A,B)}$. A secondary similarity value, $d_{(A,B)}$, was computed based on the number of IP addresses visited by both systems (d_{comm}). The d_{comm}values for all system pairs were normalized by: $d_{(A,B)} = d_{comm}/d_{max}$, where $d_{max}$is the maximum d_{comm} value across all system pairs.

Finally, they created a third similarity measure based on a Term Frequency - Inverse Document Frequency (TF-IDF) analysis of the destination address count values. For the set of destination addresses D, for each address $a_j \in D$ an IDF value was computed, quantifying the commonality of connections from source addresses $s_i \in S$. Thus, the inverse document frequency term is computed as $idf(a_j, S) = log\frac{|S|}{|\{c(s_i,a_j):s_i \in S\}|}$, where $c(a_i, s_j) = 1$ if connections were observed, and 0 if not. The terms in the vector of IDF values ($IDF_{(S,D)} = \left\{idf(a_j, S) : a_j \in D\right\}$) were used to weight the corresponding behavior vector values for each source address. IDF similarity between two systems was based on the cosine similarity of the IDF weighted behavior vectors, $IDF_{(A,B)} = cos_{(\hat{A},\hat{B})}$.

The authors compared the behavior vectors based on different intervals of time, and for different protocol (HTTP, HTTPS, and SSH) address sets, evaluating the utility of the $cos_{(A,B)}$, IDF and the (averaged) $IDF + d_{(A,B)}$ scores for identifying behavior vectors from the same user. For each test behavior vector, similarity scores were computed against the training data vectors for each source address, and the scores used to rank order the source IP addresses (by decreasing score order). For these tests, the correct source address scored highest 56% of the time for SSH connections, 26% for HTTPS and 21% for HTTP. This approach showed that there was enough consistency in the behaviors of the users that behavior vectors could be associated with the correct users at least part of the time.

Tan et al. [44] compared users based on the number of common destination addresses each host connected to. They used this similarity value to cluster networked systems at two

companies, to determine common system/user roles within the companies. The clustering method was rather unique, and depend on several parameters that had to be adjusted.

Given an enterprise network with a set of hosts (I), for each host $h \in I$ there is a set of hosts h_i connected to, $C(h_i) = \{a : a \in I\}$. If $h_i \in C(h_j)$ and $h_j \in C(h_i)$, a measure of the similarity between these hosts is given by $similarity\ (h_i, h_j) = |C(h_i) \cap C(h_j)|$. With these similarity values, relationships between the hosts in the network can be described by a neighborhood graph, nbh-graph , where each $h_i \in I$ is represented by a node . Edges between node pairs $h_i$and $h_j$are weighted with the value $similarity(h_i, h_j)$.

The nodes in a nbh-graph were initially grouped by identifying bi-connected components (BCCs) within the graph, sets of nodes connected by edges with weights greater than or equal to a value k. In a BCC, any two edges must exist in a simple cycle. Starting with a value k equal to the largest edge weight value, all nodes with edge weights greater than or equal to k (and not already grouped) are selected. BCCs within these selected nodes are identified, and the BCC is replaced by a group node, G, which inherits the connections of the nodes used to create it. Nodes claimed by more than one BCC are assigned to the largest BCC. This process is repeated using progressively lower k values. Some nodes, even with higher value edges, may be not grouped by this process. If $k < \alpha x|C(h_i)|$, where $0 \leq \alpha \leq 1$, node $h_i$is designated a group node by itself. Group nodes assume the k value under which they became groups as an attribute.

After the initial grouping, groups are iteratively merged until no further merging is possible. Groups are merged if they meet two criteria:

- The average number of connections per node within $G_1$is within β percent of the average number of connections per node within G_2
- The groups G_1 and G_2meet a similarity requirement.

The similarity requirement depends on the computation of two values to determine if similarity thresholds are met. The first , K^{max}, is the maximum group k value, or $K_{max} = max(K_{G_1}, K_{G_2})$. The second similarity value s between groups $G_1$and G_2 is determined algorithmically:

- $c1 = \frac{\sum_{h \in C(G_1)} CP(h,G_1)}{|G_1|}$, $c2 = \frac{\sum_{h \in C(G_2)} CP(h,G_2)}{|G_2|}$, $s = 0$

- For every neighboring group G' in nbh-graph that $G_1$and G_2 have in common,
  - $s = s + min\left(\frac{CP(G',G_1)}{|C(G_1)|}, \frac{CP(G',G_2)}{|C(G_2)|}\right)$
- $s = \frac{1}{2}x(\frac{s}{c_1} + \frac{s}{c_2})x100$

The CP, or connection pattern term, counts the total number of connections between a node and a group, or between groups.

The similarity requirement is met if both the similarity and K_{max}values are above set thresholds ($s \geq S^{hi}$, and $K_{max} \geq K^{hi}$), or if $K_{max} < K^{hi}$ and $s \geq S^{lo}$. The new group's k value is set to the minimum number of connections a host in the group has.

The values of α, β, K^{hi}, $S^{lo}$and $S^{hi}$are determined experimentally, to generate groups of meaningful sizes.

Flow data was collected over a day at a small (110 hosts) company and at a larger (3638 hosts) company. For the smaller company, the clustering correlated with the role structure fairly well (Rand statistic = 0.8363). Ground truth in the bigger company was not available, but the clustering was "useful and consistent" according to the network administrators. A version of this was refined in [45], where for hosts p_i and p_j the weight value becomes $w\{p_i,p_j\} = \frac{|N_{p_i} \cap N_{p_j}|}{|N_{p_i} \cup N_{p_j}|}$. This is essentially a Jaccard similarity coefficient based on the address sets, and could be used for comparing or clustering systems (or users) on the network.

Behavior Based Profiling

Profiling of network traffic to differentiate between flows created by different applications has been attempted in a variety of ways. Valenti et al. [46] distinguished available approaches as being port-based, payload-based (those doing Deep Packet Inspection (DPI) or stochastic packet inspection (e.g. extracting patterns from flows such as common string patterns, or testing the randomness of the first payload bytes)), based on statistical classification (applying data mining techniques to flow level features) and those doing behavioral classification (e.g. examining how many hosts connected with what protocol over how many ports).

Karagiannis et al. [47] developed a useful flow based construct called graphlets (see Figure 2.3). Graphlets captured flow level behavior patterns of computer systems in the form of graphs, displaying the standard five tuple flow features (source IP address, destination IP

address, source port, destination port, protocol) as vertices in a graph with edges reflecting any co-occurance of two connected features in the observed flow data records. The number of in-degree and out-degree connections of the nodes in the graphlets provide an abstraction of the relationships between the different node types. Karagiannis et al. considered high degree nodes (those nodes with more than single incoming or outgoing connections) in graphlets to be important in representing the flow activities captured. The number of in-degree or out degree counts in a graphlet thus represent dominant patterns in the captured flow data.

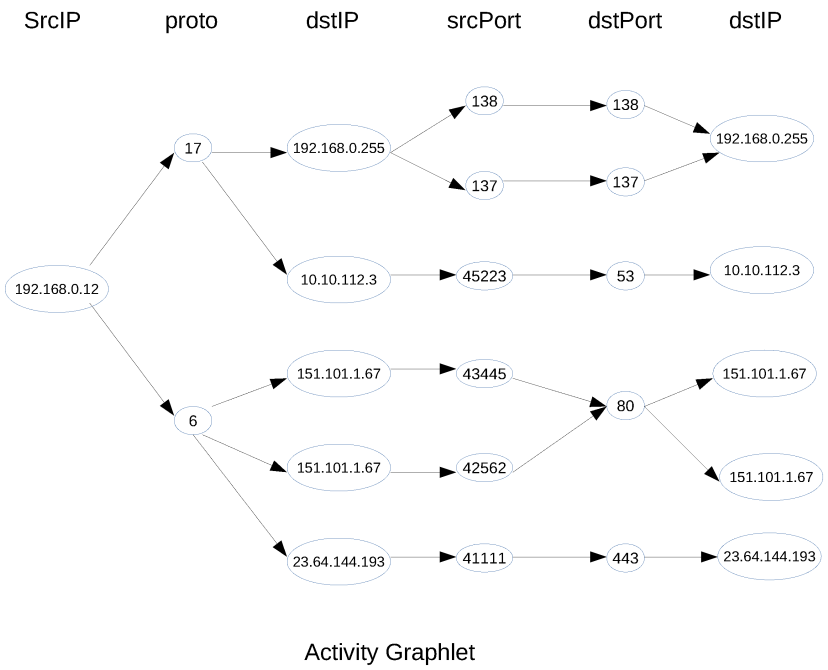


Figure 2.3. Activity Graphlet

For the purposes of our investigations, we chose to create our user profiles using interval-based statistical profiling. Most machine learning algorithms are designed to evaluate arrays or vectors of feature values for making classifications, which can be readily created based on well defined data samples.

2.5 Detection of Automatic Flows

Computers on a network are frequently communicating. Operating systems and applications often create connections to check for updates or messages, or to search for network resources.

This may happen without any human action to initiate the connections, outside of the initial opening of an application such as a web-page browser or an e-mail client. Because these connections are made automatically, they bear little relevance to understanding the activities of a user logged onto the computer. The preponderant research on automatic flows has been done in analyzing malicious programs such as bots. Bots are programs that enable the control of infected systems by others to distribute spam or to participate in large denial of service attacks.

Feily et al. [48] surveyed the forms of botnets and the programs generated to detect them. The primary characteristic that distinguishes bots from other forms of malware is the use of command-and-control channels to direct bot actions and add features. Developers of botnet detection programs have focused on this characteristic, some employing methods requiring the reading of packet data, and some based their detection methods on flow record data. [49] used packets per IP address, packets per flow, and bytes per packet metrics to compare suspected traffic to known bot models, and searched for periodic patterns by measuring the inter-flow arrival times between a client and a server, using the mean values as a fundamental period T. These values were used as inputs to train either in a hierarchical Bayesian model or a modified K-means algorithm to detect probable bot traffic. Bilge et al. [50] also used inter-flow arrival periods and flow size distributions as features for detecting bot generated flows. These approaches involve detecting repeating characteristics in the flow traffic as indications of the automated traffic flows from bots.

These repeating features contrast well with the more random feature distributions generated by human activity. Vazquez et al. [51] examined the temporal activity patterns in humans using computers, and found that the periods between actions are not Poisson distributed as had been assumed. Instead, humans tend to perform bursts of activity between long periods of inactivity, with long tailed wait time τ_w distributions ($P(\tau_w) \sim \tau_w^{-\alpha}$), where α can equal approximately 1 or 3/2 depending on the activity.

While bots are a specialized group of applications, the repetitious aspects of their behaviors versus the more random traffic patterns generated by human activities provide insight into a more generalized approach to detecting automatic flows. Because human behavior is far less predictable than the behaviors of most applications, the repetition of traffic features such as flow timing or the message sizes passed between a host and a server can be more

likely attributed to automated traffic activity than to the actions of a human. To the best of our knowledge, no research outside of our own has examined the use of Netflow features for identifying and removing automatic flow records for the purpose of enhancing analysis of user generated flows.

2.6 Conclusion

Detecting anomalous user behaviors on a network is a complex problem that needs a multitude of approaches to enable effective and scalable solutions. The availability and compactness of Netflow records make it an attractive data source for identifying anomalies, and flow metadata is an integral part of many network monitoring solutions. It makes sense that people with similar roles in an organization should be performing similar acts on the network, and if so peer behavior should provide a valid yardstick with which to evaluate individual behaviors.

If users within a role-group do not behave similarly however, grouping user data sets based on similarities observed in their data sets should be used as the preferred means of establishing group behavioral norms. The experiments described in [3] [34], [35], [38], [39], and [43] demonstrate that user network behaviors are consistent enough that traffic can be associated with users based on flow metadata. This behavioral consistency can be leveraged to identify groups of users that behave similarly, creating a behavioral yardstick based on observed user similarities.

Based on the research discussed in this chapter, the logical next step is to investigate the relationship between user roles and network behaviors based on two distinct approaches: interval based comparisons of statistical features and bottom-up clustering of Netflow based features. For this we will employ a number of different Netflow based features based on the literature and our own experimentation.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

This chapter discusses our approach to identifying and characterizing patterns of network traffic behavior, captured in the form of Netflow version 5 flow records. These patterns can be used for comparing user behaviors via statistical and machine learning techniques, and for distinguishing traffic of automatic processes from user activities.

We provide information on Netflow, the data format used to document network traffic for our analysis in Section 3.1. This section also discusses the general categories of Netflow features we reviewed to detect patterns within a set of records. Section 3.2 provides a high level overview of the pattern based approaches used in our research to identify network flow behaviors. Section 3.3 discusses the extraction and processing of the captured network data used for this research, and Section 3.4 how specific patterns observed in the collected data were discovered that indicated which flows were more likely to be generated by automatic processes. Section 3.6 covers how the attributes of flows not applicable to studying user digital behaviors were used to remove much of those flows from the data set.

3.1 Netflow Data

For the analyses in this dissertation, we primarily used Netflow version 5 records as our data source. Netflow was defined and created by Cisco initially as a means of enhancing network traffic routing [52], but as the utility of the data was recognized Cisco enabled its export for use by network administrators. While Netflow (or some variant of the standard) data is usually produced by routers or Netflow generators [53], many software tools exist to enable the capture, collection and analysis of Netflow records. One well known suite of software tools for this is the System for Internet-Level Knowledge [54], also known as SiLK, developed by the Computer Emergency Response Team (CERT) Network Situational Awareness (CERT-NetSA) Team at Carnegie-Mellon.

3.1.1 Flow Level Features

Netflow defines a flow as a one-way transfer of network data between two systems, in which the packets share a common 7-tuple (source IP address, destination IP address, source port, destination port, protocol, SNMP index of input interface, IP type of service) [55], [56]. Cisco routers (among others) utilize and can provide Netflow records with additional information such as the IP address of next hop router, or the SNMP index of the router input interface. In flow records extracted from pcap data however, not all Netflow Version 5 fields are available. Table 3.1 identifies the Netflow fields accessible from processing stored pcap data using SiLK. In Netflow v5, a flow is considered to continue between those two end points until a FIN or RST TCP flag is set, no packets have been transferred within a set time (usually 15 seconds), the flow has continued until a set timeout point (typically 30 minutes), or the router flow cache is full.

The fields in a Netflow v5 record (extracted from pcap data using SiLK) can be categorized in one of two ways. The number of packets or bytes, the flow duration and the flow start/stop epoch times are numeric values, for which comparisons such as greater than or less than have real meaning. Other values are categorical values such as IP addresses, ports, TCP flags and the protocol type (e.g. ICMP=1, TCP=6, UDP=17), for which numeric comparisons of values have no semantic meaning.

Netflow Characteristics: Netflow records summarize the data exchanges on a network, and observing these records can provide some insight into the normal characteristics of data flows. To identify these characteristics, we conducted controlled experiments where we captured Netflow data from two virtual systems (one Windows 7 and one Ubuntu Linux) during the performance of tasks a technically knowledgeable computer user would typically perform. This was performed twice for each virtual machine, once to emulate a "normal" working day and the second time to create Netflow records in a manner that would enable easier analysis.

The first Netflow data set created focused on emulating normal work activities for a period of time on each operating system (Windows: five hours, Linux: nine hours), followed by a night of continued network traffic collection while the system was idle. These activities included

Table 3.1. Netflow Version 5 Record Fields. Source: [56].

Bytes	Contents	Description	SiLK Field	Feature type	Available from pcap
0-3	srcaddr	Source IP address	sIP	categorical	Yes
4-7	dstaddr	Destination IP address	dIP	categorical	Yes
8-11	nexthop	IP address of next hop router	nhIP	categorical	No
12-13	input	SNMP index of input interface	in	categorical	No
14-15	output	SNMP index of output interface	out	categorical	No
16-19	dPkts	Packets in the flow	packets	numerical	Yes
20-23	dOctets	Total number of Layer 3 bytes in the packets of the flow	bytes	numerical	Yes
24-27	First	SysUptime at start of flow	sTime	numerical	Yes
28-31	Last	SysUptime at the time the last packet of the flow was received	eTime	numerical	Yes
32-33	srcport	TCP/UDP source port number or equivalent	sPort	categorical	Yes
34-35	dstport	TCP/UDP destination port number or equivalent	dPort	categorical	Yes
36	pad1	Unused (zero) bytes	-	n/a	No
37	tcp_flags	Cumulative OR of TCP flags	flags	categorical	Yes
38	prot	IP protocol type (for example, TCP = 6, UDP = 17)	pro	categorical	Yes
39	tos	IP type of service (ToS)	n/a	categorical	Yes
40-41	src_as	Autonomous system number of the source, either origin or peer	n/a	n/a	No
42-43	dst_as	Autonomous system number of the destination, either origin or peer	n/a	n/a	No
44	src_mask	Source address prefix mask bits	n/a	n/a	No
45	dst_mask	Destination address prefix mask bits	n/a	n/a	No
46-47	pad2	Unused (zero) bytes	-	n/a	No

web browsing, downloading new applications for installation, sending and receiving mails, opening secure shell links to servers (on Linux only), computer programming and editing documents on Windows Share drives. No attempt at separating activities was made; browsers, mail clients and Windows Share folders were left open after initial use, and normal multitasking between applications was performed. The intent was to generate Netflow records similar to that generated during normal work hours and system idle time. Activities were logged, but frequently overlapped in time. This data was labeled as the

working data set, and used as a test to validate the cleaning algorithms developed.

The second set of data was generated in a more tightly scripted manner by performing the tasks shown in Table 3.2.

Table 3.2. The Second (Scripted) Controlled Experiment

Action	Windows 7 Applications	Ubuntu 13.10 Applications
Connected to/used a Windows Share drive. Files loaded, modified, saved.	Windows Explorer	Nautilus
Opened mail client, sent/received mails	Outlook	Thunderbird
Opened SSH link	Not tested	Command line, SSH
Opened browser to www.cnn.com (HTTP)	Chrome and Internet Explorer	Chrome and Firefox
Opened browser to www.foxnews.com (HTTP)	Chrome and Internet Explorer	Chrome and Firefox
Opened browser to www.usaa.com (HTTPS)	Chrome and Internet Explorer	Chrome and Firefox
Opened browser to www.nps.edu (HTTP)	Chrome and Internet Explorer	Chrome and Firefox

Each task was separated in time (two to five minutes) from the other tasks, to enable the observation of network traffic immediately after the the task was performed, as well as the flows generated automatically during the following idle period. This approach simplified the manual labeling of the Netflow records as generated by a user or by an automatic process. All flows that could be attributed to a recent user action were labeled as user flows; all other flows were labeled as automatic. Recent actions were defined as opening a web page or starting an mail client. Flow bursts created by web pages re-loading automatically or an mail client checking with the mail server periodically for new messaged were considered automatic. This data was labeled as the scripted data set, and used primarily to determine the characteristics of automatic flows and develop algorithms to identify them. The "normal working day" flow data was used to verify the algorithms worked well against non-scripted activities.

These Netflow data sets provided us with a known "ground truth" regarding activities driving the generation of flows at specific times, which we used to examine the normal characteristics of Netflow data. For example, many of the more complex data transfers (web page loads, mail traffic) appear as bursts of flows within short spans of time. Figure 3.1 shows the flow start rates (flow starts per second) generated by our Windows 7 virtual machine over a span of approximately five and a half hours, during a period when a web-browser was left idle and began automatically reloading a web page (CNN) every 30 minutes (Figure 3.1, from approximately 5100 seconds to 19000 seconds).

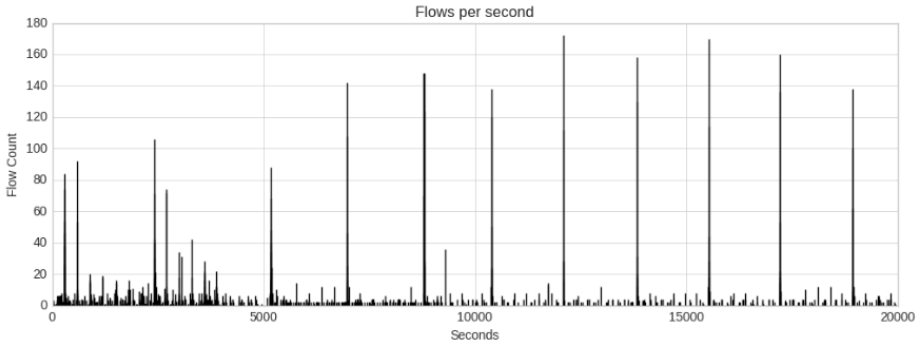


Figure 3.1. Flow Starts per Second

High flow rates per second do not necessarily translate into high data-transfer rates. Figure 3.2 shows the total megabytes per second for the same data set as Figure 3.1. While the peak data transfer rates mostly line up with the peak flow start rates, the relationship between total flows and total bytes passed is not constant. This is reflected in Figure 3.3, indicating the average flow packet sizes are mostly small, with the counts of larger packet sizes dropping off rapidly as the size increases until packet sizes approach 1200 bytes. The cluster of flows averaging between 1200 and 1400 bytes per packet represents packets associated with large file transfers. Flow generation rates, data rates and data densities (bytes per packet) are all features potentially useful in identifying and characterizing user and/or system activities on the network.

3.1.2 Netflow and User Behavior

Netflow data is a form of meta-data about network data transfers, and as with any meta-data details are lost. Using Netflow, an analyst cannot see that a user pulled a specific document from an network server. Netflow data will not trigger a "dirty word" sensor (a sensor that

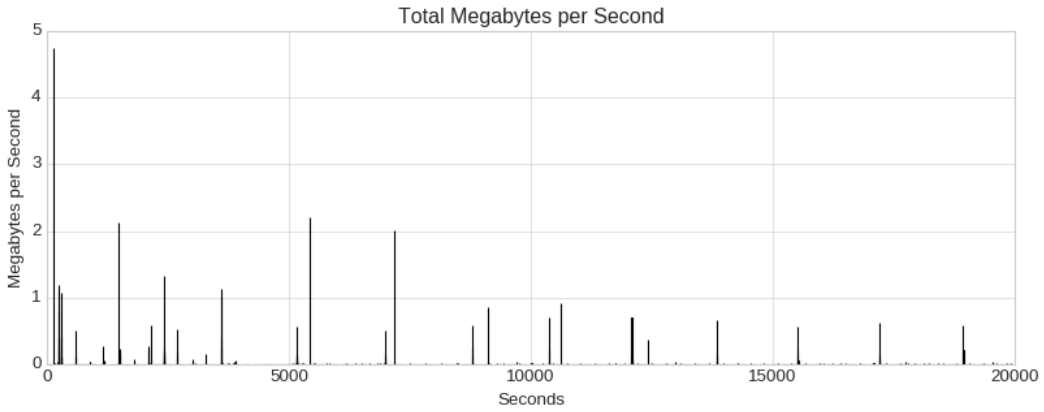


Figure 3.2. Flow Bytes per Second

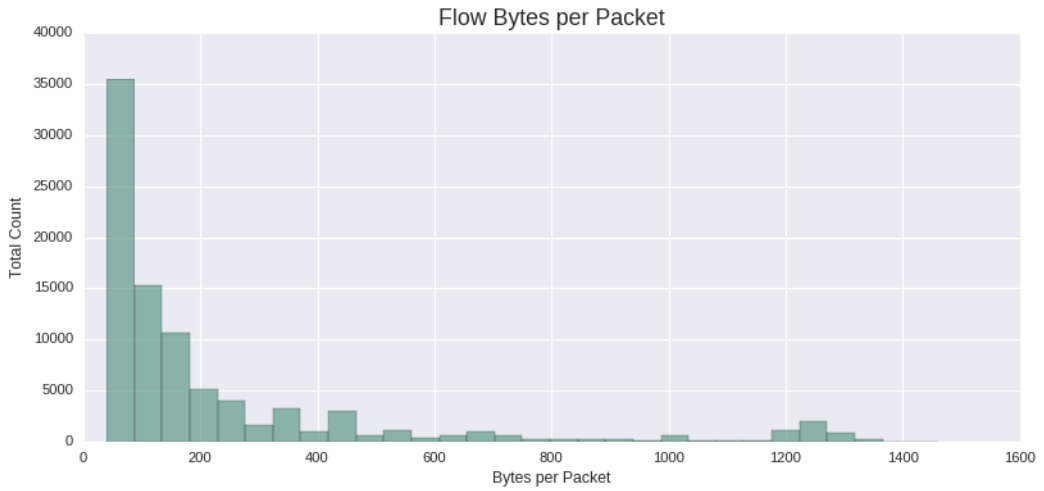


Figure 3.3. Distribution of Packet Sizes

sniffs network traffic for specific strings, like “SECRET”), or show that a user was mailing a company’s competitor. What Netflow will show is how much information was transferred and when, the IP address end points for the transfer, and some information about the kind of service used for the transfer (the protocol used, any TCP flags set during the transfer, the ports used).

With this information much can still be inferred about what the host or user was doing. The overlap in the IP addresses visited by two or more hosts can show shared communities of interest and resources. Differences in port usage can indicate differences in the level and type of network services employed by users and hosts. The timing of network flows can tell an analyst the tempo of a user’s network activity, and the network subnet accessed

when a user logs on can identify whether access was via a wireless, wired, or virtual private network (VPN) connection.

Descriptions of network traffic can be further extended by defining features describing characteristics of aggregates of flows, such as the total bytes passed to or from port 22 (SSH) or the ratio of bytes sent out over port 443 (HTTPS) to bytes received over a defined period of time. These measures are indications of the levels and types of activity a digital system was performing during the period, less precise in terms of activity than individual flow records but likely far more compact.

A set of features that correlate well with the kinds of tasks and activities a user may perform through the network may provide a basis for comparing user behaviors on the network. Anomaly detection methods based on Netflow records are widely used in Network Behavior Analysis (NBA) systems. These systems can monitor the Netflow records corresponding to hosts, servers and appliances on the network, learn normal behavior patterns, and alert when deviations are detected. These and other Netflow-based anomaly detection systems can work very well for detecting scanning, DDoS or worm behaviors [57], and have been applied to detecting botnets, peer to peer traffic, and hosts on the network generating heavy traffic. Analysis methods focused on detecting human behaviors using Netflow however, including the detection of behavioral deviations, are novel.

3.1.3 Flow Data Feature Dimensions

Netflow data captures basic parameters and measurements of data transfers over a network. From the perspective of a typical user's computer, these parameters and measurements can be put in several categories:

- Point of service: The IP addresses in the flow records can provide insight into the content of the data exchanged with the user's computer. Examples include connecting to 151.101.40.73 (www.cnn.com), and connecting to a local LAN printer.
- Type of service: The ports and protocol fields in a flow record indicate the type of services accessed by a user's computer. TCP flows to or from port 80 in most cases represent web related traffic, although use of a server port and protocol are not guarantees a particular service was accessed. Many applications (and forms of malware) use the common service ports like 80 for other purposes.

- Volume transferred: The bytes passed during a flow provide a measure of how much data was exchanged. The value includes bytes used for the packet headers.
- Flow density: Calculating flow bytes per packet provides an average packet size. Larger values would be an indication of a flow intended to pass data, while the smallest values could indicate the passing of status and connection information like TCP SYN, SYN-ACK, ACK flows, and ICMP pings.
- Flow Direction: The direction of data transfers relative to the user's system gives insight into the relationship between the connected systems. A high bytes-out to bytes-in ratio would indicate a net outgoing transfer of data, not the usual case for a client system.
- Flow Control: Netflow v5 records include the TCP flags set during a flow. TCP handshakes at the beginning or end of a series of flow exchanges and the data transfers or acknowledgments in between are more distinguishable with the TCP flag information.
- Temporal: Time plays a role in describing flow data in three basic ways:
  - Flow start time: The timestamp for the flow start-time provides context for the flow in the form of a reference point (start of the day, start of a collection interval, other flow start times).
  - Flow duration: The flow duration enables measures of flow throughput, such as milliseconds per byte (the inverse of bytes per millisecond, because flow-duration value can be zero). For computing flow statistics over defined intervals, the flow start-time and duration (or flow start-time and end-time) enable identifying flows that extend over two or more intervals.
  - Sequences: The relative positions of flows, or of flow features extracted by flows, can indicate sequences of activity on the network.

3.1.4 Categories of Netflow Features

Although the possible Netflow v5 record fields are fairly limited, the number of features that can be created based on these fields is quite large, depending on how they are combined with each other or with external data sources. We divided the potential Netflow derived features into three categories.

Direct Features: Directly generated features quantifying or describing the flow -- sIP, dIP, sPort, dPort, protocol, packets, bytes, flags, sTime, eTime -- are extracted directly from the

flow records. The duration of a flow can be determined by taking the difference between flow start and end times, but because the SiLK tool set provided this value automatically in generating flow records, for our analysis we treated duration as a direct feature.

Indirect Features: Indirect features can be derived at the individual flow-record level and can come in many forms. Indirect features can represent:

- Mathematical combinations of direct features, such as bytes per packet, packets per second, or the difference between the source and destination IP addresses (expressed as 32bit integers) to compare exchanges with local and external systems.
- Flow contextual values, where external data sources provide context for direct-feature values. Examples include:
  - Whois derived information relating to the remote address when analyzing data from a local network.
  - DNS flow query strings extracted from the pcap data. item The client system user ID, subnet ID, or operating system associated with the flow.
  - Whether packets in a flow carry a payload (which requires domain knowledge of protocols used and packet-header length).
- Flow direction relative to the local system.
- The server port used in flows between a client and a server.
- Relationships between adjacent flows. For consecutive flows in a flow set, potential features include:
  - The interval between the immediately prior flow start time and the current flow start time,
  - A measure of the address space similarity between sequential server addresses (e.g. 24 for sharing the same 24 bit prefix between IP v4 addresses, also known as a /24 subnet).

While features based on the relationships between flows could be considered to be aggregate features (Section 3.1.4), we grouped them with indirect features when the derived feature value can be expressed at the flow record level. For example, we can annotate a flow record with the interval in seconds between that flow and the previous flow in the data set.

Aggregate Features: Another category of Netflow derived features are what we term as aggregate features, which describe aggregate characteristics based on direct or indirect

features observed within a set of flow records. Aggregate features would include:

- Mathematical characteristics extracted from groups of flow records. Examples include mean and standard deviation of the flow byte values, the ratio of bytes sent or received by a client, the number of endpoints a client connects to, or the entropy of discrete feature values such as server IP addresses.
- Relative frequencies at which features occur within groups of flow records, e.g. the distribution of byte values within a set of flows. Relative frequencies can be used to rank order feature values, such as a listing of server IP addresses a client system visited or the server ports utilized, ordered by the counts, total packets or total bytes associated with each occurrence (e.g. most frequent first), enabling a comparison of sets of flow records.
- Lists of flow-feature values observed in a flow group, such as the server IP addresses. Comparisons with other flow groups can be based on the degree of overlap in the values shared between the feature-value lists.
- Lists of flow feature sequences, such as n-grams (lists of n sequential values from data) of the server IP addresses connected to by the client system.
- Features reflecting relationships between flows, such as those derived from a graphlet (Figure 2.3). The number of in-degree and out-degree connections for each graphlet node can provide some indications of the level and types of activity observed during the flow collection period.

3.1.5 Flow Set Segregation

If flow data from an large active network is collected, the records produced could be expected to describe flows spawned by a number of different systems, applications, and network-service providers (web content servers, mail servers, etc.). To evaluate the behavior of specific users or systems on a network, subsets of flows attributable to individual users must be extracted from the whole.

For our analysis, the Netflow data set (D) had to be mapped to specific users to enable comparisons. At the time of the network data collection, NPS was using the SafeConnect network-access control system ([58]) to log network events such as user logins. The records included multiple details, such as the user name and the IP address assigned to the system

used to access the network. With this data we were able to tag much of the Netflow data with user IDs, and extract the flow data for a specific user, with a specific host IP address and period of time that user was logged on. For multi-user systems (such as in classroom labs), a user's association with flows to/from a system started upon system logon, and ended when the next person logged in.

Thus for each user ID in the data set ($u_i \in U$) a subset of flow records in the captured data was identified ($D[u_i]$). If the user employed more than one digital device on the target networks, $D[u_i]$ included flows to/from more than one client system. For any analysis performed on $D[u_i]$, the flow data was further subdivided based on the assigned IP address of the system used. To describe this, a flow subset tied to a user u_i interacting with a system at IP address c_IP_j can be expressed as $D[u_i][c\_ip_j]$. Specifying use of a specific protocol (pr_k), server port (sp_l) and distant end IP address (e_ip_m) can be expressed as $D[u_i][c\_ip_j][pr_k][sp_l][e\_ip_m]$.

3.2 Patterns Within Flow Sets

We define patterns within a set of Netflow records as the repeated occurrence of one or more features (direct, indirect or aggregate) within a data set in a manner that appears non-random. Examples of this include the same or similar feature values (e.g. flows sharing the same server port, protocol, packets, bytes and flag values) occurring with greater frequency than other values for those features observed within the data, or sequences of feature values repeating with greater frequency than other sequences. This same determination of non-randomness can be applied to features used to characterize aggregates of Netflow records, where the relative rates of occurrence of one or more features or repeats of feature sequences can indicate some favored modes of flow activity.

For identifying patterns between different data sets, the degree of overlap in the same or similar categorical feature values between two sets can be used as a similarity metric. Vector differences of various kinds can provide similarity metrics for numeric features. Much of machine learning is based on comparing data sets, to determine how they are related. Clustering and classification algorithms compare data instances based on similarity, applying a distance metric or rule set to group similar instances together and provide decision boundaries between dissimilar instances. For our analysis, we applied machine-learning

algorithms to our data sets to determine how well the patterns they identify in the data sets map well to the roles of the users associated with the data.

3.2.1 Direct and Indirect Feature Patterns

In a set of flows exchanged between a client and one or more servers, the presence of one or more direct or indirect features repeating with greater frequency than others can be an indication of automatic network activity. We examined the relative frequencies of bidirectional flows to evaluate how well they indicate automatic activity.

For each flow subset sharing a common user, client IP address, protocol, server port, and endpoint IP address ($D[u_i][c\_ip_j][pr_k][sp_l][e\_ip_m]$) we identified sequential flow pairs, matching each flow with the response (if present) from the other system. We then converted the bidirectional flows into vectors: the server port, protocol, packets, bytes, flag values from flow one and server port, protocol, packets, bytes, flag values from flow two. If no matching return flow was observed, default values (zero for numerical fields, an empty string for flags) were added. We then counted the relative frequency of value vectors derived from the subset. We refer to these flow-value vectors as flow signatures.

We also investigated relative frequencies of the intervals between flow starts as an indicator of automatic activity. For a given flow record subset defined by shared user, client IP address, protocol, server port and end-point IP address ($D[u_i][c\_ip_j][pr_k][sp_l][e\_ip_m]$) feature values, we can take a count of the intervals between flow start times (rounded to the nearest second) for each direction of flow. Intervals reflective of bursts of traffic are not counted (e.g. intervals of less than one second), because at that point most flow-interval values fall within the rounding window.

3.2.2 Aggregate-Feature Patterns

The features of individual flows provide fairly fine grained views on the activities of a computer user and the computer being used. Groups of flows provide more comprehensive views of network activities, particularly when the flows in the group are not narrowly defined (i.e. using flows drawn from exchanges between one client and multiple servers, rather than one client and one server).

For our research we used a standard approach to identifying and defining groups of flow records, i.e. groupings based on flows having start times within defined sampling periods (e.g. 15 minutes). The flows generated during this period can be characterized by aggregate features, or features about aggregates of individual flow features. Once grouped we can create distance or similarity measures based on the aggregate features selected, providing a means of comparison between different samples of flow records. Potential aggregate features can come in a number of forms; we focused on two categories: aggregate statistical measures and rank ordering of feature values.

Aggregate Statistical Measures: Statistical descriptors of a flow set can include:

- Standard statistical measures (count, mean, mode, first and third quartiles, minimum, maximum, standard deviation, etc.) of numerical direct or indirect Netflow features (e.g. bytes, packets, duration)
- Aggregate measures dependent on flow direction (e.g. total bytes or packets in and out, fraction of flows inbound, ratio of bytes in to bytes out)
- Protocol dependent measures (fraction of TCP flows with no payload, fraction of TCP flows with SYN flag set, number of broadcast flows, etc.)
- Information theory measures: Entropy is a measure of the randomness of a set of discrete (bytes, packets, IP addresses, ports, protocol) values. Entropy is computed over a set of values ($x_i \in X$) as $H(X) = -\sum_{x_i \in X} p(x_i) log(p(x_i))$, where $p(x_i)$ is the probability of the value x_i occurring within X.
- Distributions of direct or indirect feature values. Examples include the distribution (histogram) of flow byte values, TCP flag values, or interflow intervals (rounded to some multiple of seconds) observed within a flow set. Distributions can also be made of aggregate-feature values such as the mean byte values within flow samples taken every 15 minutes, over the period of a day. A disadvantage of distributions is that they provide a larger amount of data than statistical measures like mean and standard deviation.

Of these, feature value distributions provide additional dimensions relative to single value statistical measures, and thus can be more descriptive of how direct or indirect flow feature values vary within a flow set. On the other hand, standard statistical measures such as mean and standard deviation or information theory measures such as entropy are succinct. For the

same set of direct or indirect Netflow features, a concatenation of feature value distributions can result in much longer value vectors to describe the same data set.

Rank Ordering of Feature Values: Categorical Netflow features (IP addresses, ports, protocols) observed within a flow data set can be expressed as sorted lists. Examples include the sorting of port and protocol values based on the number of flows, number of bytes, or number of packets passed via that port and protocol. The port and protocol combination associated with the highest total value (e.g. total bytes), would be listed first, followed by the second highest, and so on. A variation of this aggregate feature that we used in our work is the Port Priority Vector (PPV).

Port Priority Vectors: If the flow records derived from multiple users are extracted and the observed tuples of source port and source protocol, and destination port and destination protocol are sorted, this sequence of port and protocol values can be used to create an index list for comparing specific users and system activities. Sorting (what we call "rank order") is by associated measures (total flows, packets or bytes) of each observed port and protocol combination, where we order the port-protocol pairs by decreasing measured values. Let $P_* = [p, p_2, ..., p_m]$ be a vector of rank-ordered port-protocol values derived from flows associated with the activities of a group of users, $U = \{u_1, u_2, ..., u_n\}$. Let $P_i = [q_1, q_2, ..., q_o]$ be the rank-ordered port-protocol values based on the activities of user $u_i \in U$. We can define a mapping function, $r(q_j)$, which provides the index value of port, protocol q_j in P_*. With this mapping function, we can create a Port Priority Vector, $PPV_i = [r(q_1), r(q_2), ..., r(q_o)]$, which is a list of indices mapping the ports in P_i to their positions in P_*. This concept is shown in Figure 3.4, where the rank-ordered port, protocol (TCP=6, UDP=17) combinations (e.g. port 80 traffic using UDP protocol, or 80, 6 in Figure 3.4) from a user's flow data are compared to the rank-ordered list from a larger group of users, to create a Port Priority Vector.

Using the Port Priority Vector (PPV) concept, we can quickly see if a user's system is significantly employing unusual server ports (as high index values are listed near front of the PPV), or is employing server ports in proportions similar to the other users (Figure 3.5). Like feature value distributions, PPVs are more descriptive than lists of single Netflow feature values because they provide a comparison between data from an individual user and the average behavior of a larger population.

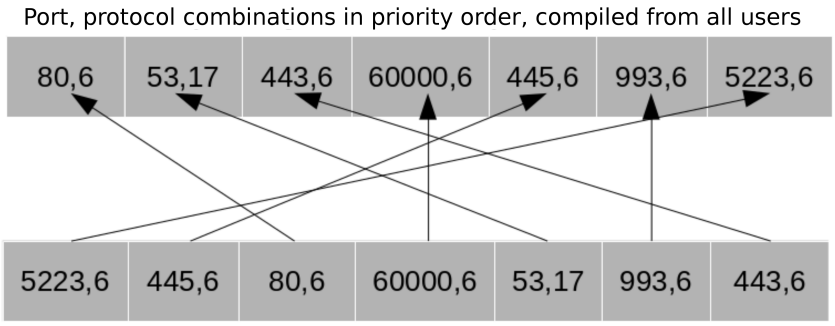
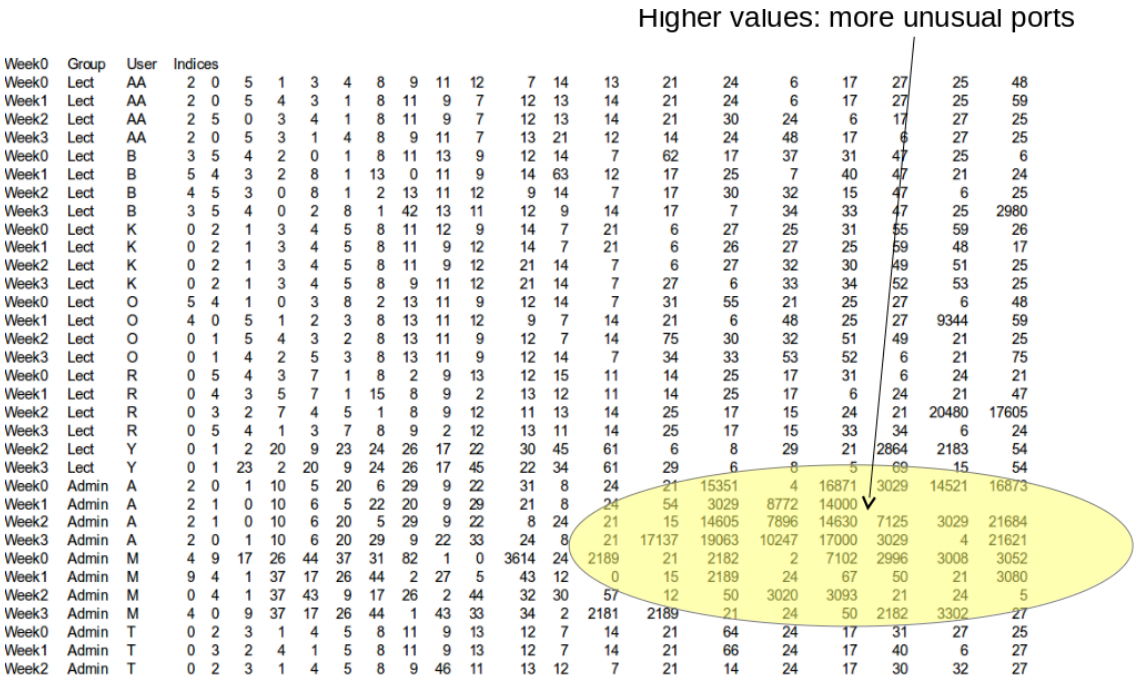


Figure 3.4. Port Priority Vectors



Week0	Group	User	Indices																		
Week0	Lect	AA	2	0	5	1	3	4	8	9	11	12	7	14	13	21	24	6	17	27	25	48
Week1	Lect	AA	2	0	5	4	3	1	8	11	9	7	12	13	14	21	24	6	17	27	25	59
Week2	Lect	AA	2	5	0	3	4	1	8	11	9	7	12	13	14	21	30	24	6	17	27	25
Week3	Lect	AA	2	0	5	3	1	4	8	9	11	7	13	21	12	14	24	48	17	6	27	25
Week0	Lect	B	3	5	4	2	0	1	8	11	13	9	12	14	7	62	17	37	31	47	25	6
Week1	Lect	B	5	4	3	2	8	1	13	0	11	9	14	63	12	17	25	7	40	47	21	24
Week2	Lect	B	4	5	3	0	8	1	2	13	11	12	9	14	7	17	30	32	15	47	6	25
Week3	Lect	B	3	5	4	0	2	8	1	42	13	11	12	9	14	17	7	34	33	47	25	2980
Week0	Lect	K	0	2	1	3	4	5	8	11	12	9	14	7	21	6	27	25	31	55	59	26
Week1	Lect	K	0	2	1	3	4	5	8	11	9	12	14	7	21	6	26	27	25	59	48	17
Week2	Lect	K	0	2	1	3	4	5	8	11	9	12	21	14	7	6	27	32	30	49	51	25
Week3	Lect	K	0	2	1	3	4	5	8	9	11	12	21	14	7	27	6	33	34	52	53	25
Week0	Lect	O	5	4	1	0	3	8	2	13	11	9	12	14	7	31	55	21	25	27	6	48
Week1	Lect	O	4	0	5	1	2	3	8	13	11	12	9	7	14	21	6	48	25	27	9344	59
Week2	Lect	O	0	1	5	4	3	2	8	13	11	9	12	7	14	75	30	32	51	49	21	25
Week3	Lect	O	0	1	4	2	5	3	8	13	11	9	12	14	7	34	33	53	52	6	21	75
Week0	Lect	R	0	5	4	3	7	1	8	2	9	13	12	15	11	14	25	17	31	6	24	21
Week1	Lect	R	0	4	3	5	7	1	15	8	9	2	13	12	11	14	25	17	6	24	21	47
Week2	Lect	R	0	3	2	7	4	5	1	8	9	12	11	13	14	25	17	15	24	21	20480	17605
Week3	Lect	R	0	5	4	1	3	7	8	9	2	12	13	11	14	25	17	15	33	34	6	24
Week2	Lect	Y	0	1	2	20	9	23	24	26	17	22	30	45	61	6	8	29	21	2864	2183	54
Week3	Lect	Y	0	1	23	2	20	9	24	26	17	45	22	34	61	29	6	8	5	69	15	54
Week0	Admin	A	2	0	1	10	5	20	6	29	9	22	31	8	24	21	15351	4	16871	3029	14521	16873
Week1	Admin	A	2	1	0	10	6	5	22	20	9	29	21	8	24	54	3029	8772	14000			
Week2	Admin	A	2	1	0	10	6	20	5	29	9	22	8	24	21	15	14605	7896	14630	7125	3029	21684
Week3	Admin	A	2	0	1	10	6	20	29	9	22	33	24	8	21	17137	19063	10247	17000	3029	4	21621
Week0	Admin	M	4	9	17	26	44	37	31	82	1	0	3614	24	2189	21	2182	2	7102	2996	3008	3052
Week1	Admin	M	9	4	1	37	17	26	44	2	27	5	43	12	0	15	2189	24	67	50	21	3080
Week2	Admin	M	0	4	1	37	43	9	17	26	2	44	32	30	57	12	50	3020	3093	21	24	5
Week3	Admin	M	4	0	9	37	17	26	44	1	43	33	34	2	2181	2189	21	24	50	2182	3302	27
Week0	Admin	T	0	2	3	1	4	5	8	11	9	13	12	7	14	21	64	24	17	31	27	25
Week1	Admin	T	0	3	2	4	1	5	8	11	9	13	12	7	14	21	66	24	17	40	6	27
Week2	Admin	T	0	2	3	1	4	5	8	9	46	11	13	12	7	21	14	24	17	30	32	27

Figure 3.5. PPV Example

3.3 Data Collection

3.3.1 Campus Data Collection

The Netflow records used for this research came from packet capture files collected from a large campus academic building over a five-week period, February 3rd to March 9th, 2013. The captured network traffic was converted into Netflow records using the SiLK [54] software tool set. Altogether over 1.162×10^9 Netflow records were captured.

The network traffic was collected from a spanning port on a switch serving a building with four academic departments, hundreds of people, as well as dozens of classrooms and computer labs. Most systems attached to the building's wired infrastructure connected under a single /21 IPv4 subnet, and a wireless system occupied a /20 subnet address spaces. In addition, a /21 subnet for VPN connection traffic was visible in the data. A total of 2985 unique internal IP addresses were present in the traffic traces from the wired, wireless and VPN networks. This number does not equal the total number of systems connected on these networks, however. As most network IP addresses were assigned using DHCP, during the observation period a system may use one or more IP addresses, and IP addresses may be assigned (at different times) to more than one system.

To correlate users on the network with their group affiliations, we accessed the SafeConnect network access-control data used by the NPS Information Technology and Communications Services (ITACS) office. The ITACS database system automatically collects system information when a user connects to the network, including user name, the host hardware (media access control, or MAC) address, operating system, and IP address. This information for each system was correlated with organizational data available on the campus Lightweight Directory Access Protocol (LDAP) server, to associate user titles and departments with the user logon records. For systems employed by multiple users, we identified the points in time when different users logged on to the systems, and used these points to identify windows of time during which each specific user could be associated with the system. After scrubbing the logon data we could identify users that could be classified under the roles listed in Table 3.3.

Confidentiality of user information in captured network traffic was ensured through several methods. First, the maximum packet size during capture was limited to 100 bytes; any

Table 3.3. Counts of the Role Groups in Data Set

Categories	Role Groups	Count
Staff	Administration	29
	Admin	30
	Class management	8
	Funding/acquisition	12
	IT support	35
Faculty	Lecturer	42
	Research Assistant	84
	Tenure	151
Student	Distance Learning Student	16
	Masters	954
	PhD Student	12
Unclassified	Unclassified	208

additional data was discarded. This eliminated much of the packet data content from being recorded, and reduced the total storage requirements needed for the data. Second, captured packet data was stored in encrypted form, to eliminate inadvertent leakage. Finally, after the initial analysis of the captured network data and the attribution of user roles, identifying data that could be used to associate specific user names to network traffic was deleted. From that point on, user-identification numbers were used to track individuals within each group, decoupling individual users (and potential privacy issues) from the Netflow records.

The host operating system types were captured by implants ITACS mandated be installed on Windows/Apple computers connecting to the NPS network. The observed systems used on the target networks were primarily Apple (581), Windows 7 (1151), and Windows XP (3346), with 55 Windows Vista and 71 Windows 8 systems reported as well (Table 3.4). Linux based operating systems were not well documented in the logon data, as SafeConnect did not have implants for Linux systems: they were primarily detected as hosts for Microsoft Windows virtual machines that connected to the network. The operating system breakout listed in Table 3.4 represents the total recorded during the collection period for the entire campus. Not all these systems detected were observed in our collected traffic data.

Because each major operating system generates automatic flow patterns characteristic to that operating system, we limited the data set we evaluated to flows generated by Microsoft Windows based systems. Originally we tried combining data from the Apple systems

Table 3.4. Counts of the Operating Systems in the Data Set

MacIntel	84
OSX 10	496
iPad	1
Vista	55
WIN7	1151
WIN8	71
WINXP	334
UNKNOWN	27
Win32	184
Win64	1
Linux	19

with that of the Windows systems, but found there were too many differences in regard to software and protocols to enable meaningful generalizations. When we correlated our feature sets against both our role groups and the different operating system types in the data, the strongest correlations found were to specific operating systems rather than to user groups.

3.4 Pre-Processing Analysis

Netflow provides a valuable means of monitoring network status. Using Netflow records, a network administrator can identify issues including systems and application that hog the available bandwidth, DDoS attacks, and routing problems. To use Netflow as a means of monitoring user behavior however, it makes sense to isolate the Netflow records that reflect user activities. In many cases, records have little to no bearing on what users are doing with their computers on the network. We evaluated the flow records we collected at NPS, and to the extent possible removed those that did not appear to reflect the activities of users on the wired, wireless and VPN subnets we identified.

3.4.1 Non-Algorithmic Analysis

The network traffic used for our research was extracted from a spanning port on an NPS router servicing the Computer Science/International Studies/Operations Research (Glasgow) building on campus. Traffic collected from the tap included all traffic to and from

Glasgow, as well as data exchanges transiting the router. To get a sense of the kinds of traffic within the data capture (see Figure 3.6), we evaluated the Netflow data in terms of:

- The subnet end-points of each flow.
- The TCP/UDP server ports used
- The total flows being passed per day

Several categories of traffic not relevant to our analysis of user behaviors were identified (Figure 3.6).

- On 11-12 February 2013 traffic to/from one server on the network dominated the flows generated during that period. The server intensely port scanned numerous systems on the network, sending TCP SYN and UDP packets across a number of common service ports as well as selected port ranges for each scanned system. These scans contributed approximately 27% of the total flows observed during the collection period.
- During 3-25 February 2013, a large portion of the collected data flows were attempted (outgoing SYN packets only) connections from a small number (11) of internal systems over port 5223. Port 5223 is associated with Apple Push Notification traffic; whois queries of the target IP addresses verified the attempted connections were to servers owned by Apple.
- A significant portion of the captured flow traffic did not have any endpoint within the subnets our analysis was focused on; these flows were labeled as external.
- We evaluated the ports used in in the data set. Ports 67 and 68 (DHCP), plus port 123 (NTP), contributed to the total flow count in the data but had little relevance to evaluating user caused traffic patterns as obtaining or renewing system IP addresses or syncing to network time are typically actions performed automatically.

The flows associated with these observations were flagged for removal, which significantly reduced the total number of flows to analyze. Table 3.5 shows a breakout of the categories of flows determined to be not relevant to our analysis.

3.4.2 Feature Pattern Based Analysis

An important characteristic of normal traffic is that a significant fraction of the recorded flows can correspond to automatic system activities, i.e. the flows were not directly ini-

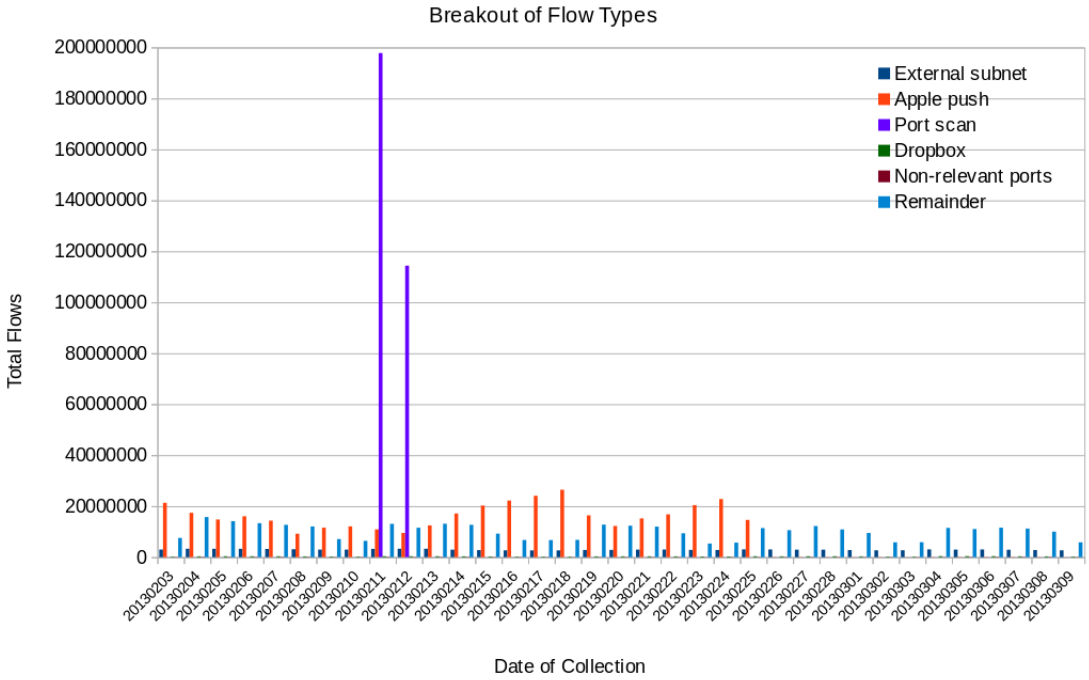


Figure 3.6. First Flow Analysis Breakout

Table 3.5. Non-Relevant Flows in Our Data

Name	Total flows
All flows	1.162×10^9
Scanner flows	3.122×10^8
Apple Push Notification	3.79×10^8
External flows	1.032×10^8
Ports 67, 68, 123	2.11×10^6

tiated by a user action. Examples include mail clients checking with mail servers for new messages, applications and operating systems polling repository servers for updates, applications checking the local network for available services they can connect to, etc. These automatic flows can pose a problem for systems using Netflow for evaluating user behaviors, as they add noise to the data. Complicating user behavior analysis further, different applications and operating systems exhibit different patterns of automatic flows. For example, older Microsoft operating systems and applications typically employ ports and protocols associated with NetBIOS over IP (137/UDP, 138/UDP, 139/TCP) far more than Linux based systems. Many Apple computers use the port/protocols 5223/TCP, 2195/TCP and 2196/TCP to support the Apple Push Notification Service, a push protocol for updating

data on some Apple applications. For enterprise networks hosting a variety of user computer configurations, similarity measurements between individual or groups of users could be skewed by flows tied to the system configurations used to access the network. Some of these configuration-based differences can be filtered out by enumerating configuration dependent flow types and filtering these out of the data, but at the expense of losing potentially valuable information. Many of the flows associated with specific operating systems are automatically generated, and can be removed from data sets if correctly identified.

Timing: Recognizing flows created by automated processes is a necessary step before removing them from the flow data. Vazquez et al. [51] found that for web browsing and other computer use activities, people tend to perform bursts of activity between long periods of inactivity. The distribution of wait times between human actions was long tailed ($P(\tau_w) \sim \tau_w^{-1}$). Programs behave less randomly than humans. Karasaridis et al. [49] used repeating interflow arrival periods as an indicator of bot traffic. Bilge et al. [50] also used interflow arrival periods and flow-size distributions as features for detecting bot-generated flows. Such repeating characteristics in the flow traffic provided indications of the automated traffic flows from bots. Bartlett et al. [59] used Haar wavelets to detect network flows recurring over regular periods, including BitTorrent control messages, RSS feed aggregators polling for updates, keyloggers, operating-system updates, and other automatic flow activities.

Figure 3.7 shows the distribution of intervals (in seconds) between flow start times for the port 8443/TCP (SafeConnect) flows in our test data. As the figure shows, the most common interval values were about 60 seconds. The second most common interval was less than one second (flow-burst activity), followed by intervals of 30 seconds between flow starts. Note these have far higher counts than the background activity and so are easy to recognize.

We examined the timing behavior of subsets of the scripted flow data, each subset defined by sharing the same source and destination IP addresses, server port, protocol, and endpoint IP address values ($D[u_i][c\_ip_j][pr_k][sp_l][e\_ip_m]$). Figure 3.8 shows distributions of interflow interval values, rounded to the nearest second, for selected flow subsets. It is apparent that for some flow exchanges between a server and a client some interflow interval values greater than one second are far more common than other values. While not all client-server interactions exhibited high interval counts (as shown in the port 53 plots), it was fairly

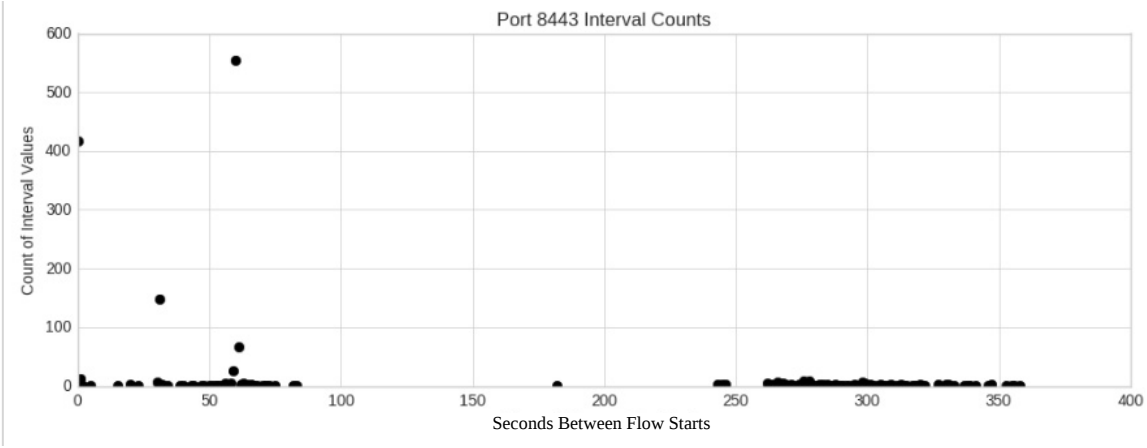


Figure 3.7. Repeating Interflow Intervals

common to find high counts for interval values close to some multiple of 15 seconds.

Based on our hypothesis that repeated patterns indicate automatic flows, we developed an algorithm to identify flows following intervals with high count values as automatic. For each client IP address in a user data set ($D[u_i][c\_ip_j]$) , flows were divided into subgroups sharing the same protocol, server port and distant end IP address ($D[u_i][c\_ip_j][pr_k][sp_l][e\_ip_m]$). Each subgroup was sorted chronologically by flow start times, and divided again based on flow direction (to or from the local system). For each flow direction, the interval values between flow start times were rounded to the nearest second. In most cases for a large flow set the dominant interval values were low (0-1 second), due to the number of bidirectional flow and response exchanges. To filter out these, we ignored interflow interval values of less than two seconds. The occurrence of each interval value was counted, and outlier values were identified via the Tukey outlier algorithm described in Section 3.4.2. Flows identified as occurring immediately after outlier interval values were flagged as automatic.

Bidirectional Flow Vectors: Repeating features occur when an application must fulfill a set of functions that would need to be repeated periodically (i.e. checking for updates or services). Applications can generate both automated and user-initiated flows, depending on whether a human is actively using the application (e.g. generating mails via an mail client) or if the application is left idle (e.g. the mail client polling the mail server for new messages). In such a case we would expect that bidirectional data exchanges between a client and server would include repeated queries and responses, with corresponding repeated byte and packet

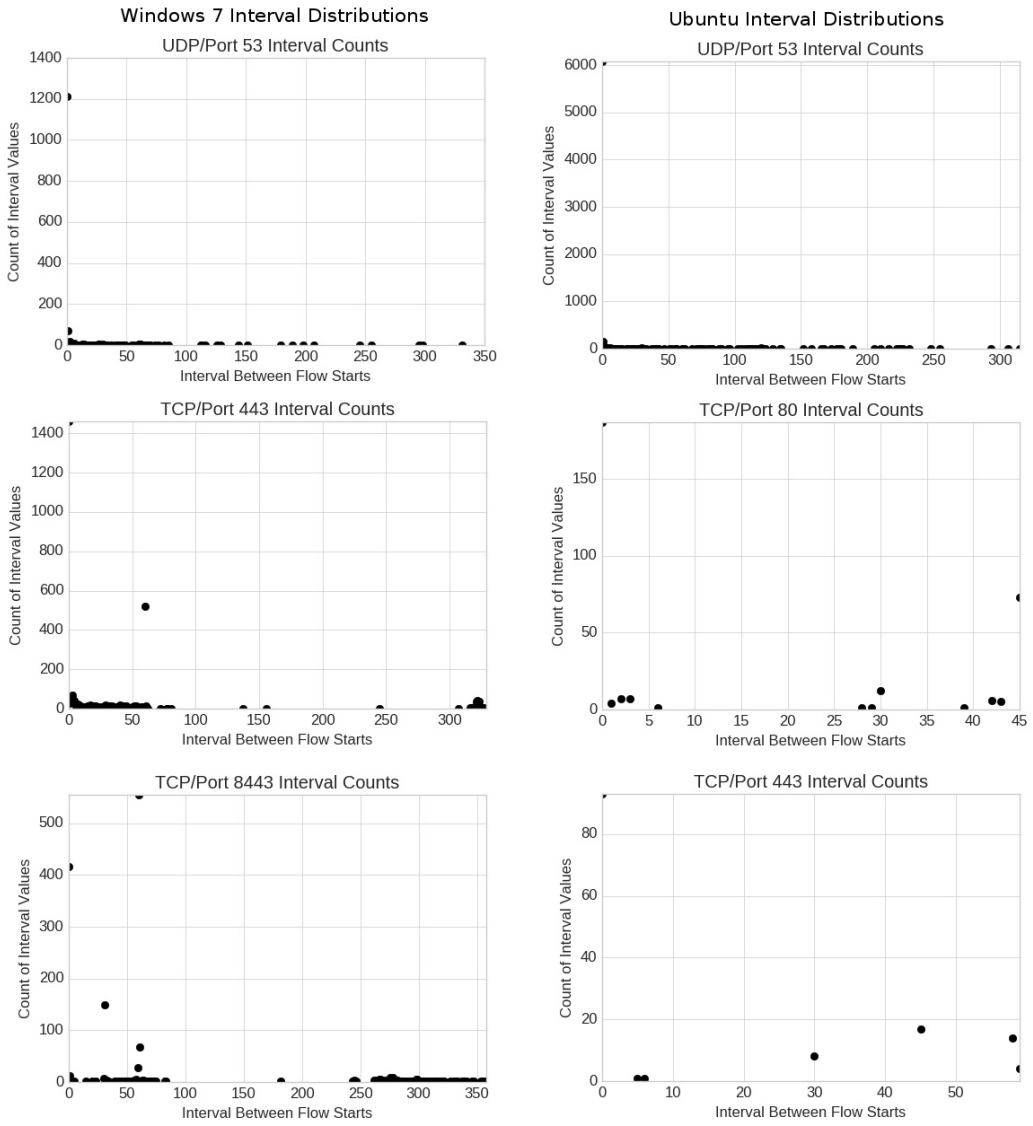


Figure 3.8. Example Interval Distributions

count values for each exchange.

These bidirectional exchanges can be codified as signatures, and counted to determine which exchanges occur more frequently than others. For our analyses, a bidirectional flow signature consists of the server port, protocol, outgoing packets, outgoing number of bytes, outgoing TCP flags, incoming number of packets, incoming number of bytes and incoming TCP flag values of a bidirectional flow. Figure 3.9 plots the count values (ordered by

decreasing count values) for signatures representing four services observed in our test data, DNS (port 53/UDP), HTTP (port 80/TCP), HTTPS (Port 443/TCP) and flows created by a SafeConnect [60] implant (port 8443/TCP). SafeConnect implants on host systems were used by NPS to monitor user network accesses, and ensure connected systems had up to date antivirus programs running. We can see that some bidirectional flow signatures occur far more frequently than others. The repeated signatures are indications of repeated actions.

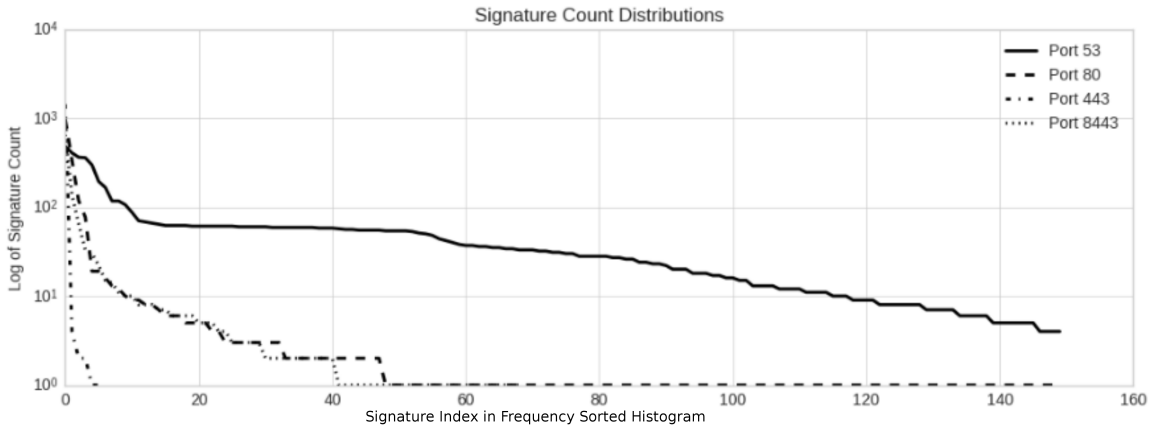


Figure 3.9. Bidirectional Flow Distributions

We examined the frequency at which bidirectional flow pairs (flow between a client and server, and a response flow back) occurred within the flows collected from the Windows 7 and Ubuntu virtual machines. To count flow pair incidences, we selected flow subsets sharing the same feature values ($D[u_i][c\_ip_j][pr_k][sp_l][e\_ip_m]$). Each subset was sorted chronologically, and we defined flow pair signatures as value vectors consisting of the flow server port, protocol, number of packets, number of bytes and flag values of an initial flow concatenated with the return flow packets, bytes and flags values. If no return flow was found, default values for the return flow features (zero packets, zero bytes, no flags) were inserted. If flow start times in the flow pair appeared equal, we assumed the client to server flow occurred first when creating the signature.

Figure 3.10 shows the relative frequencies at which the same flow pair signatures occur within the flow subsets. The counts of each signature observed within a subset are sorted in descending order to create a distribution-like list of values (usually exponential in appearance), which is used to provide a sense of how quickly the signature counts drop off. The different lines in each graph represent signature distributions from flows to different distant

end servers. While the many flow subsets demonstrate some high count flow pair signature instances, in most cases the great majority of flow pair signatures exhibit very low count values.

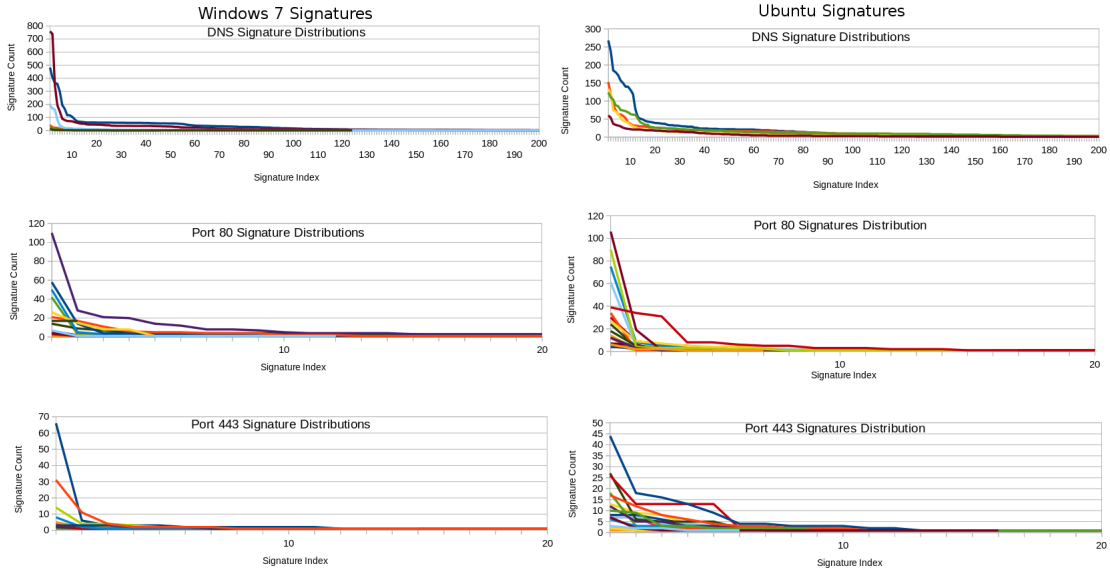


Figure 3.10. Per Server Signature Pseudo-Distributions

Again, based on our assertion that highly repeated patterns indicate automatic activity, we developed an algorithm that identifies flows associated with high count signatures as being automatically generated. For each $D[u_i][c\_ip_j]$, flow-pair signature vectors were generated using the approach described above and each flow-pair signature was counted. Outlier count values were identified via the Tukey outlier algorithm described in Section 3.4.2. Flows with outlier counts were flagged as being automatic.

We also evaluated the relative frequencies of signatures derived from a client communicating with multiple servers, by studying the flow subsets sharing the same client IP address, server port and protocol values ($D[u_i][c\_ip_j][pr_k][sp_l]$). Figure 3.11 shows the pseudo-distributions of the signatures found per subset in the Netflow data. Upon selecting flows flagged based on outlier signature counts, the selected records primarily consisted of TCP handshake (SYN, FIN, or RST) related flows. This is consistent with evaluating signature frequencies across the same server port and protocol values, as TCP handshake signatures would be shared across all completed TCP client-server interactions. TCP handshakes were common across both user generated and automatic flow sets however, and so searching for

outlier signature counts from this perspective was not useful for differentiating the two sets.

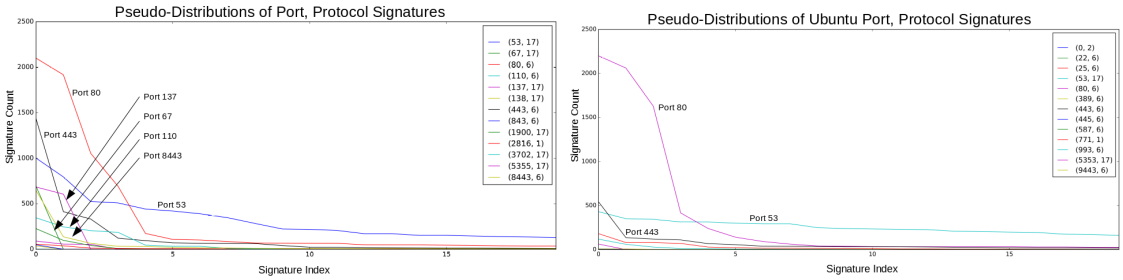


Figure 3.11. Per Port/Protocol Pseudo-Distributions

Flow Sequences: To observe flow patterns of traffic known to be automatically generated, it is especially helpful to examine the flows produced by the virtual machines when the systems were not being actively used. Figure 3.12 shows a representative extraction of flow records when the Windows 7 system was idle. Two characteristics are apparent; the first being that flow sequences repeated (as shown by color coded sections), and the second characteristic was that the sequences appeared to be separated by larger interval gaps between flow start times.

Sequence features were often not exact repeats: note the mail related sequences highlighted in light blue; where the numbers of packets and bytes do not match exactly for the third and fourth flows. This means that a similarity threshold is needed to define how similar flows in two sequences must be to declare a match. Another necessary threshold is the gap size between flow starts required to separate two sequences. This gap threshold is less critical when evaluating flow sequences within flows sharing the same client and server IP addresses, server port and protocol, rather than evaluating sequences between a client and two or more servers. In most cases this increases the average time between discrete sequence starts to much more than one second. This was the approach we took, and we used one second as our interval threshold.

We examined the relative frequencies at which similar sequences appear in the data set to identify outlier instance counts, by grouping sequences based on shared source IP address, destination IP address, protocol, packet count and TCP flag sequence values and similar flow byte values. For each $D[u_i][c\_ip_j]$, we divided flows into subgroups sharing the same protocol, server port and distant end IP address ($D[u_i][c\_ip_j][pr_k][sp_l][e\_ip_m]$). Subgroup

sIP	dIP	sPort	dPort	pro	packets	bytes	flags	sTime	duration	eTime	Interval	Server owner
10.0.2.15	204.102.114.49	61835	80	6	6	758	FSPA	2014/04/14T21:41:07.397	0.187	2014/04/14T21:41:07.584	10.019	Akamai Technologies
204.102.114.49	10.0.2.15	80	61835	6	6	2557	FSPA	2014/04/14T21:41:07.397	0.187	2014/04/14T21:41:07.584	0	Akamai Technologies
10.0.2.15	205.155.65.20	61836	443	6	7	729	FSPA	2014/04/14T21:41:07.871	0.353	2014/04/14T21:41:08.224	0.474	www.nps.edu
205.155.65.20	10.0.2.15	443	61836	6	9	3210	FSPA	2014/04/14T21:41:07.871	0.353	2014/04/14T21:41:08.224	0	www.nps.edu
172.20.24.130	10.0.2.15	443	61837	6	20	6007	SPA	2014/04/14T21:41:08.166	3.452	2014/04/14T21:41:11.618	0.295	NPS e-mail
10.0.2.15	172.20.24.130	61837	443	6	15	6262	SRPA	2014/04/14T21:41:08.166	3.452	2014/04/14T21:41:11.618	0	NPS e-mail
10.0.2.15	10.0.2.255	137	137	17	3	234		2014/04/14T21:41:08.783	1.499	2014/04/14T21:41:10.282	0.617	Internet Assigned Numbers Authority
10.0.2.15	204.102.114.49	61839	80	6	52	3138	FSPA	2014/04/14T21:41:09.397	0.698	2014/04/14T21:41:10.095	0.614	Akamai Technologies
204.102.114.49	10.0.2.15	80	61839	6	93	115812	FSPA	2014/04/14T21:41:09.397	0.698	2014/04/14T21:41:10.095	0	Akamai Technologies
10.0.2.15	172.20.24.130	61841	80	6	3	152	S	2014/04/14T21:41:11.621	9.007	2014/04/14T21:41:20.628	2.224	NPS e-mail
10.0.2.15	204.102.114.49	61842	80	6	6	758	FSPA	2014/04/14T21:41:11.646	0.179	2014/04/14T21:41:11.825	0.025	Akamai Technologies
204.102.114.49	10.0.2.15	80	61842	6	6	2557	FSPA	2014/04/14T21:41:11.646	0.179	2014/04/14T21:41:11.825	0	Akamai Technologies
10.0.2.15	204.102.114.49	61844	80	6	112	8098	FSPA	2014/04/14T21:41:13.648	8.228	2014/04/14T21:41:21.876	2.002	Akamai Technologies
204.102.114.49	10.0.2.15	80	61844	6	202	248486	FSPA	2014/04/14T21:41:13.648	8.228	2014/04/14T21:41:21.876	0	Akamai Technologies
10.0.2.15	205.155.65.20	61849	443	6	7	729	FSPA	2014/04/14T21:41:21.611	0.35	2014/04/14T21:41:21.961	7.963	www.nps.edu
205.155.65.20	10.0.2.15	443	61849	6	9	3210	FSPA	2014/04/14T21:41:21.611	0.35	2014/04/14T21:41:21.961	0	www.nps.edu
172.20.24.130	10.0.2.15	443	61850	6	20	5975	SPA	2014/04/14T21:41:21.907	1.246	2014/04/14T21:41:23.153	0.296	NPS e-mail
10.0.2.15	172.20.24.130	61850	443	6	14	6190	SRPA	2014/04/14T21:41:21.907	1.246	2014/04/14T21:41:23.153	0	NPS e-mail
10.0.2.15	172.20.24.130	61851	80	6	3	152	S	2014/04/14T21:41:23.155	9.001	2014/04/14T21:41:32.156	1.248	NPS e-mail

Figure 3.12. Repeating Idle Sequences

flows were further subdivided into sequences based on the intervals between flow start times, where intervals between flow starts within a sequence had to be less than a threshold θ_s. Intervals between flows greater than θ_s demarcated the end of one sequence and the start of another.

To group similar sequences together for comparison, for each sequence of length >2 the flow records were sorted in turn by the flow packet count, TCP flag, and source IP values. Flow start times were recorded to the closest millisecond; this step ordered the flow sequences independently of the flow start-time stamps as many sequences contain flows with the same start time values. The ordered lists of the sequence source IP address, destination IP address, protocol, packet count and TCP flag values are concatenated and hashed, and the flow sequences are then grouped based on hash values. At this point, sequences sharing the same hash value could only differ in byte and temporal (flow start, end and duration) values.

For each sequence group (sharing the same hash value), let $S = \{s_1, s_2, ...s_n\}$ represent the identified sequences. Let $Q = \emptyset$, an empty sequence set to hold sequences dropped from S if required. Let M be a byte value matrix, where the bytes in row i are byte values for the ith row for each sequence in S and the jth column represents the ordered byte values of sequence s_j. For each row i in M, let μ_i be the mean byte value and σ_i the standard deviation. Let θ_b be the byte value similarity threshold, such that if $\theta_i/\mu_i > \theta_b$, one or more byte values in row i are too different. The byte value in row i farthest from μ_i is determined and all sequences in S with that byte value in flow i are extracted and placed in Q. This process is repeated until $\theta_i/\mu_i > \theta_b$ for each row of the sequences in S. The sequence set Q containing the outlier byte values is added to the sets still to be evaluated.

After all sequence groups are evaluated and regrouped (if necessary) for similarity, the number of sequences in each group is counted. For sequence of three or more flows (longer than the bidirectional flows tested in the flow feature vector method), very few sequences repeated frequently enough to be identified as outliers. Of those that were identified, 90%+ of the flows were also flagged by the timing and bidirectional signature algorithms as being automatic. Because of this and the high processing cost of this algorithm, this approach was not used to clean the data.

Web Page Reloads: Another form of repeating sequences was observed if a web browser was left open on certain websites (e.g. www.cnn.com, www.foxnews.com) while the system was left unattended. For these websites, pages would reload automatically at set intervals (~30 minutes for CNN, ~10 minutes for Fox News). Figure 3.13 shows a representation of this behavior, by graphically plotting the observed number of flows per second generated while a browser was left on www.cnn.com. As the figure shows, the web page reloads showed similar, but not identical, flow counts in loading the web page. An examination of the flow data showed that the reload flow sequences were not identical either, but share similar amounts of data transfer with an overlap in the IP addresses connected to.

Based on these observations, plus the intuition that automatic flows would be characterized by repeating characteristics, we developed and tested detection algorithms for identifying automatic web-page reloads.

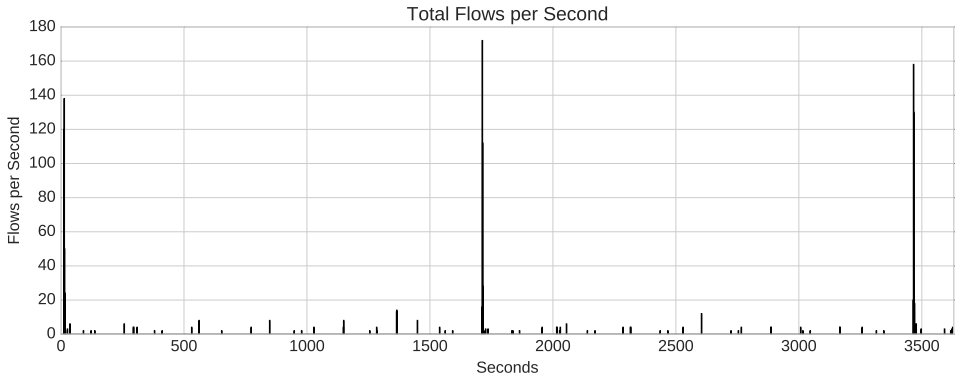


Figure 3.13. Web Page Reload Flow Rates for CNN on Chrome Browser

Depending on the web-page site, page loads can result in numerous flows created in rapid succession. HTTP, HTTPS and DNS (ports 80, 443 and 53) usually constitute the majority

of the flows, although other server ports can be found in the page load records. To identify web-page reloads, we set the following criteria for tagging web page load activity:

- Web page loads were preceded and succeeded by intervals (τ) between flow starts greater than threshold θ_s, or $\tau \geq \theta_s$ (see Section 3.4.2 for discussion of thresholds).
- Excluding DNS and TCP flows with no payload, the number of flows (n) in a web page loads was greater than threshold θ_l, or $n \geq \theta_l$.
- The fraction of the bytes transferred via HTTP or HTTPS connections (f) was greater than threshold θ_w, or $f \geq \theta_w$.

To identify web page reload flow sets, we extracted flow subsets that met our web-page load criteria and then grouped them based on similarity. Flow sets F_1 and F_2 were considered to be similar if they:

- Did not differ in the number of flows by greater than threshold θ_c, or $min(|F_1|/|F_2|, |F_2|/|F_1|) \leq \theta_c$.
- Had similar byte-transfer distributions for both server IP addresses and server ports.

To measure the similarity of byte-transfer distributions, let the number of bytes passed between the client and a server at IP address a_i in flow sets F_1 and F_2 be $b(F_1[a_i])$ and $b(F_2[a_i])$, respectively, where $b(x)$ represents the number of bytes passed by the flows in x. The default value of $b(F_x[a_i])$ is zero, if a_i is not in F_x. For each a_i in the combined flows of F_1 and F_2, byte values were normalized using $m_{ip} = max(b(F_1[a_i]), b(F_2[a_i]))$, and the Euclidean distance between sets of normalized values measured. The distance based on per-IP address byte transfers to and from m servers is then $d_{ip} = ((\sum_{j=1}^{m} (\frac{b(F_1[a_i])}{m_{ip}} - \frac{b(F_2[a_i])}{m_{ip}})^2)^{1/2})/m$.

Likewise, we define the number of bytes passed between the client and the servers over server port p_j in flow sets F_1 and F_2 as $b(F_1[p_j])$ and $b(F_2[p_j])$, respectively. Byte values passed between the client and server were normalized using $m_p = max(b(F_1[p_j]), b(F_2[p_j]))$, and the Euclidean distance between sets of normalized values measured. The distance based on per-port byte transfers over n server ports is then $d_p = ((\sum_{j=1}^{n} (\frac{b(F_1[p_j])}{m_p} - \frac{b(F_2[p_j])}{m_p})^2)^{1/2})/n$. The total distance between the flow sets is then $d = \frac{d_{ip}+d_p}{2}$, or the average of the differences for IP address and server port byte transfers. We required the distance between flow sets to be less than the threshold θ_d for them to be grouped together.

Figure 3.14 shows web page load subsets as boxes, with i_k the most common interval observed between subset starts. The blue boxes represent a set ($\mathcal{F}$) of similar web page loads. Within a group of similar page load flow sets, the intervals between flow set start times was captured. The interval values were rounded by values proportional to the interval length. For each interval I between web-page loads, a rounding value $d = It_{delta}$ was computed.. Because even shorter web page reload intervals (~10 minutes) could vary by 10 seconds or more, the rounding values (d) were set to the nearest multiple of 10 seconds. Each interval value was rounded as $I' = d \bullet \left\lfloor ((I + 0.5d) \bullet d^{-1}) \right\rfloor$. The rounded interval values in I' are counted, and web page load sequences where two or more sequences in a row follow an outlier (rounded) interval value are labeled as automatic.

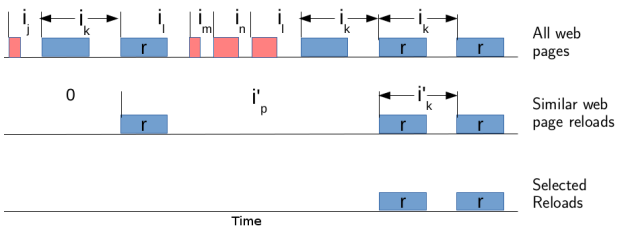


Figure 3.14. Web Reload Selection

Because automatic flows are the result of varied applications and operating-system functions, it should be expected that automatic flows should exhibit only some of the repeating features identified in the test data. This means each detection algorithm emphasized precision, not recall on the test data. Put together however, the detection algorithms should have good recall (a "set covering" approach)

Threshold Testing

The threshold values for the algorithms used to clean data collected from the campus network was determined via grid testing. For each algorithm, threshold values were varied while applying the algorithms to our labeled test data (Section 3.1.1). Thresholds were set to achieve precision values greater than or equal to 0.95 for each algorithm. Precision was given a higher priority than recall for cleaning the data, as we did not want to discard potentially valuable information.

Outlier Detection Threshold: The algorithms developed to detect automatic flows employ several thresholds that can be set to determine if flows are automatic or user generated.

A threshold common to the signature, timing, and web reload algorithms is the outlier detection threshold θ_o, where for a set of count values sorted in increasing value, we determine outlier counts via application of the Tukey [61] box plot creation algorithm:

Compute the InterQuartile Range: $IQR = 3^{rd}\ quartile\ value - 1^{st}\ quartile\ value$

Identify the high outlier values: $C_o = \{c_j^{'} \geq 3^{rd} quartile\ value + \theta_o \times IQR \mid c_j^{'} \in C\}$.

We tested the effect of varying the value of θ_o between 1.0 and 2.0 in increments of 0.1 for the cleaning algorithms that employed the outlier detection method. Our criteria for an acceptable result was one where the precision of the selection process was greater than 0.95. For our experiments, θ_o was set to 1.5.

Web Reload Thresholds: Several threshold values are associated with identifying web-page reloads. Besides using an outlier threshold θ_o and a sequence-interval threshold θ_s, the algorithm also requires thresholds on identifying flow groups as web-page loads, grouping similar web-load flow sets together for testing for repeated interval values, and a value rounding factor for comparing inter-page load intervals. The values enclosed in square brackets in the next paragraph were the values we used for our analysis.

Flow groups were defined as consecutive flows with intervals between flow start times $\leq \theta_s$ [4 seconds]. Web page loads were identified as flow groups where most of the bytes passed were HTTP(S) related (the ratio of total bytes passed via ports 80 and 443 to total flow group bytes was $\geq \theta_w$ [0.9]), and the flow group was large enough (number of flows in the group was $\geq \theta_l$ [20 non-DNS or empty-payload flows]. To ensure any interval analysis was performed on related web page loads, for flow sets a and b the flow lengths needed to be similar ($||a| - |b|| \leq \theta_c * min(|a|, |b|)$ [θ_c = 0.25]. In addition, flow sets a and b needed to be similar in terms of port usage and IP addresses visited. The distance algorithm and threshold, θ_d [0.9], is described in Section 3.4.2. Our factor τ_{delta} for rounding off interval values between web page loads was set to 0.08. These values were determined using a grid search for best recall with maximum precision.

3.5 Cleaning Data

This section discusses how the algorithms discussed in Section 3.4 was applied to the data.

3.5.1 Non-Algorithmic Cleaning

Based on our evaluation of the captured traffic (Section 3.4.1), the first cleaning phase consisted of removing flows from the data that were irrelevant to our analysis. Flows to/from IP addresses not within the subnets selected for analysis were dropped, as were network-security port-scan related flows. Flows to and from ports 5223 (Apple Push Notification), ports 67 and 68 (DHCP), and 123 (NTP) were also removed as irrelevant to user behavior analysis. While the port-based record removals could have been performed using the automatic flow-detection algorithms we developed, removing them up front greatly reduced the subsequent number of flow records to process as shown in Table 3.5.

The remaining flows after this filtering process were then evaluated using algorithmic processes for detecting automatic flows. Analyses comparing role-group data sets in Chapter 4 were performed using "cleaned data" and "unfiltered data", to measure the improvement (if found) in classifier performances provided by the cleaning process. The unfiltered data referenced in Chapter 4 refers to flow data in which the irrelevant flow records had been removed, but for which the remainder have not yet been cleaned using the algorithms discussed in Section 3.4.2 and in Section 3.5.2.

3.5.2 Algorithmic cleaning

For each of the patterns identified in Section 3.4.2, algorithms were written to identify flows matching those patterns. Flow data for each user ID within our role defined groups (Table 3.3) was extracted, and separated into flow subsets based on which IP address associated with that user ID ($D[u_i][c\_ip_j]$) from the ITEC logon database was present. Flow subsets from non-Microsoft Windows based systems were discarded, and each remaining flow subset was tested for the presence of automatic flows.

Timing: For each $D[u_i][c\_ip_j]$, flows were divided into subgroups sharing the same protocol, server port and distant end IP address ($D[u_i][c\_ip_j][pr_k][sp_l][e\_ip_m]$). Each subgroup was sorted chronologically by flow start times, and divided again based on flow direction (to or from the local system). Interval values between flow start times were rounded to the nearest second, and outlier counts of interval values greater than two seconds identified via the Tukey outlier algorithm described in Section 3.4.2. Flows identified as occurring immediately after outlier interval values were flagged as automatic.

Bidirectional Flow Vectors: For each $D[u_i][c\_ip_j]$, flows were divided into subgroups sharing the same protocol, server port and distant end IP address ($D[u_i][c\_ip_j][pr_k][sp_l][e\_ip_m]$). Each subgroup was sorted chronologically by flow start times, flow pair signature vectors were generated using the approach described in Section 3.4.2 and each unique flow pair signature counted. Outlier count values were identified via the Tukey outlier algorithm described in Section 3.4.2. Flows identified as corresponding to flow signatures with outlier counts were flagged as being automatic.

Web Page Reloads: For each $D[u_i][c\_ip_j]$, flows were divided into subgroups based on the intervals between flow start times, where each subgroup was preceded and succeeded by intervals $\geq \theta_s$. Flow subgroups representing web page loads were identified and grouped based on flow feature similarities as described in Section 3.4.2. The intervals between sequential, similar web page loads were rounded proportionally (i.e. longer intervals meant larger rounding values), and the rounded interval values counted. Outlier counts of interval values were identified via the Tukey outlier algorithm described in Section 3.4.2.For instances where web pages were found to reload for two or more times following outlier interval values, the flows associated with the web reload events were marked as automatic.

3.6 Comparing User Groups

3.6.1 Comparisons Through Classifiers

The relationship between a user's organizational role and the Netflow data they generate was tested through the use of two classifier algorithms, a nearest centroid classifier (also known as the nearest mean classifier) and a support vector machine (SVM) [62]. A nearest centroid classifier determines the mean feature values (centroid) for each class in a training set of data. For a given set of training feature-vectors and class labels $\{(\overrightarrow{x_1}, y_1), ..., (\overrightarrow{x_i}, y_i), ..., (\overrightarrow{x_n}, y_n)\}$, the mean value for each feature is computed per class ($\overrightarrow{\mu_k} = \frac{1}{|C_k|} \sum_{j \in C_k} \overrightarrow{x_j}$), where C_k is the set of y_i index values and y_i= class k. Histogram-based vectors are treated in the same manner, where each position in the vector is considered a feature. Once the class centroids are defined, test vectors are classified based on the closest class centroid. Nearest centroid classifiers are linear discriminators, providing a simple test on the separability of the test vectors and can be used to differentiate between multiple classes. We used the nearest

centroid module in the scikit-learn Python library [63] to provide the classifier for these tests.

Support Vector Machines (SVMs) identify class boundaries or hyperplanes that provide the largest margin between data points comprising the different classes. For classification problems where the data points are not linearly separable, SVMs can use a non-linear kernel function to map data points into a higher dimensional space where the data points may be separable. SVMs are inherently binary classifiers, discriminating between two classes of data. For multi-class classification problems, they can be used in a "one versus the rest" approach, where for n classes n classifiers are created, with each classifier trained to separate the data from one class from the rest of the data set. They can also be used in a "one against one" approach, where for n classes $n * (n-1)/2$ classifiers are created, with each classifier trained to separate the data from two classes. For our tests we used SVMs with a radial-basis-function (rbf) kernel, to enable testing for non-linear boundaries between the different classes. The SVMs were trained in a "one versus the rest" classification approach. For our experiments, we used the SVC module in the scikit-learn Python library [63].

To test identification of user roles we created control data sets including data vectors from each of the role groups. For each classification test we randomly selected users from each role group and extracted without replacement their associated feature vectors. For n groups, the fraction of total feature vectors removed from each group was set at $100 * n/(n+1) \pm 2\%$. These selected users and their data were used to create a pseudo-role group, so named because the group was created to be role-neutral.

If the classifiers can discriminate data of role-group A from that of other role groups, it would mean that some portion of the feature vectors in role-group A are more similar to each other than to vectors from the other role groups. Creating a group that consists of a mixture of data extracted proportionately from the true-role groups would blend those features unique to each role group. If user roles do have a measurable effect on the Netflow records each user generates, the recall or precision for either classifier in correctly identifying members of the pseudo-role group should be consistently worse as compared to the valid role groups.

3.6.2 Comparisons Through Clustering

To test whether feature vectors associated with users in the same role group were inherently similar, we clustered feature vectors using the K-means++ algorithm. The number of clusters (k) was set to 50, to provide enough cluster centers for smaller groups of similar clusters to emerge. If clusters are found with highly uneven representation by the different role groups (i.e. the cluster membership is dominated by one or two role groups), this would indicate that some subset of flow patterns was associated primarily with those one or two groups.

3.6.3 Comparisons By Users

While classification algorithms can be used to test how separable feature data is for a few classes, they are less useful when comparing data sets associated with hundreds of users. To compare the differences between users within the role-groups, we performed a pairwise comparison between the users in our role groups based on the feature vectors associated with each user.

Let $U = \{u_1, ..., u_i, ..., u_m\}$ be the set of users identified for our research, and $V_i = \{v_1, ..., v_k, ..., v_n\}$ a set of feature vectors associated with a user u_i. For each $u_i \in U$ we group the feature vectors by week of the sampling interval used to generate them and for each vector group we compute the mean values for each feature in the vector group to create a centroid vector. Let V_i' be the set of centroid vectors created from the feature-vector-data set of user u_i. For each u_i and $u_j \in U$, let d_{ij} represent the mean of the pairwise euclidean distances between the centroid vectors in V_i' and V_j'. For m users, this produces an $m \times m$ matrix of distance values, D.

Let $G = \{g_1, ..., g_k, ..., g_p\}$, where g_k is one of the defined role groups each containing a subset of the users. For each subset of users, we collected three types of distance-value distributions. The first distribution focused on the self-similarity of feature vectors produced by the same user. For each user u_i in group g_k we collected the d_{ii} distance values, or the mean-pairwise distance between the centroid vectors in v_i. This set of self-similarity distances provide a measure of how similar centroid vectors generated by the same user are over time.

The second distance distribution focused on the mean distances between the centroid vectors

of pairs of users in group g_k. For each user $u_i, u_j \in g_k$ where $i \neq j$, we collected the d_{ij} distances. This set of distances show how similar centroid vectors generated by users in the same role group are.

The third type of distance distributions focused on the mean pairwise distances between centroid vectors generated by users in group g_k and the feature vectors generated by users in other role groups. For each role group g_l, where $l \neq k$, we collected mean pairwise distances d_{ij} between each user $u_i \in g_k$ and user $u_j \in g_l$. This set of distances show how similar the centroid vectors generated by users in one role group are to centroid vectors generated by users in other role groups.

Through these comparisons we can determine if user flow patterns are consistent over time (i.e. self-similar), and if user flow patterns are more similar to those of users in their own role group than they are to users in other role groups.

CHAPTER 4:
Design of Experiments

This chapter describes the experiments used to evaluate whether users behave far more as individuals, with their own patterns of behavior when accessing enterprise networks, than they behave as members sharing common tasks and behaviors associated with an organizational role. The impact of a user's role on computer usage is examined based on comparing Netflow-derived aggregate features, extracted from network traffic generated by users associated with different role groups. To reduce the potential impact of operating-system-specific flow behaviors affecting the features extracted from the data, we limited the flow data used for our analyses to that from hosts running a version of Microsoft Windows. Experiments were performed twice, using flow-record data both before and after data cleaning (i.e. removing flows identified as being automatically generated), to measure the impact of that cleaning of the data.

In designing the experiments testing the relationship between user roles and the Netflow records they generate, we first defined four sets of Netflow-derived features to compare. We then extracted the feature data sets from the records associated with each user in our defined role-groups. For each of the feature data sets, we extracted a subset of users and their associated flow data from each role-group to create an additional, artificial group, which we designated as the pseudo group. We used the term pseudo group because unlike the data sets extracted from one of our identified role-groups, this artificial group was designed to be role neutral. It is therefore a false, or pseudo, role-group created for testing the impact of roles in our flow data sets.

Using these feature vectors, we tested them using two different classifier algorithms, to determine if the classifiers could differentiate feature vectors derived from data associated with the original role groups better than those of the pseudo group. In addition, we clustered the feature vector data sets using K-means++ to determine if portions of the data from the same role group will cluster together. Finally, we performed a pairwise comparison all the identified users based on the mean distances between each user's feature vector data sets, determining how similar each user's feature vectors are to themselves, to the feature vectors

of other users in their role group, and to the vectors of users in the other role groups. The results of these tests are discussed in Chapter 5.

4.1 Feature Definitions

To test the level of relationship between the roles of users in an organization and the characteristics of the Netflow records they produce, we generated four derived data sets.

- The first derived data set consisted of vectors of port-protocol volume measures (fraction of total bytes in, fraction of total bytes out, standard deviation of byte values) for selected port-protocol combinations (Table 4.3) plus aggregate statistical and information theory based features. The complete list of features for this data set is provided in Table 4.4. This data set was used to test how well user roles could be matched with a diverse set of aggregate Netflow derived features.
- The second derived data set consisted of a subset (Port Behavior) of the features listed in Table 4.4, tested separately from the other feature types. This port-protocol oriented feature set was tested by itself to compare classifier results with two other (third and fourth data sets) representations of port-protocol usage.
- The third derived data set consisted of pairs of byte value distributions (explained in more detail in Section 4.1.3). For each of the port-protocol combinations in Table 4.3, byte-value distributions are created for each direction of flow, and the pairs of byte-value distributions concatenated into vectors.
- The fourth derived data set incorporated Port Priority Vectors (Section 3.2.2), which are composed of indexed references to a consolidated (based on all user data) ordered listing of port and protocol usages. PPVs provide additional context to user port and protocol usage relative to that global norm.

4.1.1 Baseline Feature Set

Research into identifying anomalous host behaviors via machine learning techniques and Netflow data has often employed statistical measures of well-known ports as features. Frias-Martinez [3] created profiles using standard deviation and mean values of the number of hosts connected to, the number of packets and the bytes per packet for ports 21, 22, 25 and 80. In another experiment, Frias-Martinez created host system profiles based on the total

number of flows, average bytes per flow, average bytes per packet, average flow durations, total packets, average packets per flow and total unique IP addresses connected to using ports 80 and 22. For each approach, profiles were clustered to group similar hosts together, and the feature ranges per cluster used to determine the normality of new feature values. While this approach was successful in detecting synthetically generated attacks (those that involved introducing profiles with outlier feature values), limiting the feature set to a few well known ports ignored much of the available Netflow data that can be used for profiling.

To identify the most potentially useful features for differentiating role-based groups, we first examined the flow traffic captured over the five week collection period. Table 4.1 shows the 20 most frequently observed ports (source or destination) and protocols for each role group in the data prior to cleaning, listed in descending order based on flow counts. Also listed are the top 20 ports and protocols for all the traffic. Table 4.2 show the top 20 ports and protocols found after cleaning the data.

One difference between the two tables that can be seen is that Table 4.2 shows more dynamic ports listed, in the range used by Windows services (MS Dynamic RPC range, ports 49152-65535 for MS Server 2008, Windows Vista and later versions). With the elimination of many flows not user generated (automatic), ports associated with flows containing less repetitive patterns become more prominent. SSH traffic, for example, emerges in the top 20 ports/protocols used by PhD students in the cleaned data, while traffic to/from port 1900 (Universal Plug N' Play, an automatic process) became less prevalent.

Based on the server ports observed in the cleaned flow data, we chose the port-protocol combinations listed in Table 4.3 for generating port based flow features. Port values of zero (observed in non-TCP/UDP flows), ICMP type/code values (Netflow encodes these values as 256 x ICMP type + ICMP code), port 8443 (used by the SafeConnect system employed by NPS) and port 123 (NTP) were not used to create features.

For each of the selected top-server-port-protocol (p^2) combinations listed in Table 4.3 we defined measures based on the total bytes out ($\overrightarrow{b_{p^2}}$), total bytes in ($\overleftarrow{b_{p^2}}$), the mean byte value passed (μ_{p^2}), the standard deviation of the byte values passed (σ_{b^2}), as well as the total bytes passed ($|b|$) during each measured interval for each user. From these values, we defined features reflecting the fraction of bytes passed out ($\overrightarrow{b_{p^2}}/\,|b|$), the fraction of bytes passed in

Table 4.1. Top Port-Protocol Combinations Observed Before Cleaning.

Admin		Admini-stration		Class mgmt		DL Student		Funding/ acq		IT support		Lecturer	
Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot
80	TCP	53	UDP	80	TCP	80	TCP	60001	TCP	137	UDP	53	UDP
443	TCP	137	UDP	137	UDP	443	TCP	80	TCP	0	IGMP	80	TCP
137	UDP	80	TCP	139	TCP	53	UDP	137	UDP	389	UDP	137	UDP
53	UDP	389	UDP	443	TCP	5222	TCP	0	IGMP	80	TCP	389	UDP
0	IGMP	0	IGMP	0	IGMP	60000	TCP	443	TCP	443	TCP	0	IGMP
445	TCP	443	TCP	2524	TCP	137	UDP	60000	TCP	445	TCP	443	TCP
60000	TCP	8080	TCP	53	UDP	5355	UDP	53	UDP	53	UDP	8080	TCP
60001	TCP	9100	TCP	60001	TCP	0	IGMP	445	TCP	60000	TCP	60000	TCP
389	TCP	8055	TCP	60000	TCP	8443	TCP	389	TCP	389	TCP	445	TCP
88	TCP	60000	TCP	2668	TCP	1900	UDP	138	UDP	1720	TCP	389	TCP
8443	TCP	1900	UDP	8443	TCP	88	TCP	88	TCP	57621	UDP	138	UDP
135	TCP	445	TCP	389	TCP	50165	TCP	8443	TCP	88	TCP	88	TCP
138	UDP	138	UDP	0	ICMP	3702	UDP	8014	TCP	9443	TCP	8055	TCP
0	ICMP	8443	TCP	445	TCP	63949	TCP	389	UDP	5222	TCP	8443	TCP
139	TCP	161	UDP	135	TCP	49900	TCP	49155	TCP	8443	TCP	8014	TCP
49155	TCP	0	ICMP	2967	TCP	49361	TCP	135	TCP	49443	TCP	61087	TCP
389	UDP	771	ICMP	49155	TCP	53171	TCP	12174	TCP	8080	TCP	61084	TCP
8014	TCP	389	TCP	138	UDP	5353	UDP	5355	UDP	138	UDP	49155	TCP
9443	TCP	88	TCP	5222	TCP	49375	TCP	0	ICMP	26822	TCP	49330	TCP
5353	UDP	8014	TCP	2048	ICMP	53622	TCP	771	ICMP	49503	TCP	135	TCP

Masters Student		PhD Student		Program Mgmt		Research Asst		Tenure		overall	
Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot
80	TCP	80	TCP	137	UDP	80	TCP	137	UDP	80	TCP
0	IGMP	53	UDP	0	IGMP	137	UDP	80	TCP	137	UDP
137	UDP	443	TCP	3283	TCP	0	IGMP	443	TCP	0	IGMP
53	UDP	60000	TCP	53	UDP	53	UDP	0	IGMP	53	UDP
8080	TCP	5222	TCP	445	TCP	443	TCP	53	UDP	443	TCP
443	TCP	137	UDP	80	TCP	445	TCP	8080	TCP	8080	TCP
389	UDP	8443	TCP	389	TCP	60000	TCP	389	UDP	389	UDP
445	TCP	993	TCP	138	UDP	9443	TCP	445	TCP	445	TCP
9443	TCP	0	IGMP	88	TCP	389	TCP	9443	TCP	60000	TCP
8055	TCP	49238	TCP	8014	TCP	8443	TCP	60000	TCP	9443	TCP
389	TCP	49240	TCP	389	UDP	60001	TCP	389	TCP	389	TCP
138	UDP	161	UDP	443	TCP	138	UDP	8055	TCP	60001	TCP
88	TCP	445	TCP	49155	TCP	135	TCP	138	UDP	138	UDP
8014	TCP	138	UDP	135	TCP	2967	TCP	8443	TCP	88	TCP
60000	TCP	5355	UDP	5355	UDP	3571	TCP	88	TCP	8055	TCP
49159	TCP	88	TCP	50497	TCP	88	TCP	0	ICMP	8443	TCP
8443	TCP	902	TCP	0	ICMP	49155	TCP	4385	TCP	8014	TCP
135	TCP	1900	UDP	33355	UDP	0	ICMP	135	TCP	135	TCP
49155	TCP	49211	TCP	771	ICMP	389	UDP	49155	TCP	0	ICMP
0	TCP	49197	TCP	49351	TCP	2232	TCP	10019	UDP	49155	TCP

Table 4.2. Top port-protocol Combinations Observed After Cleaning

Admin		Admini-stration		Class mgmt		DL Student		Funding/ acq		IT support		Lecturer	
Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot
80	TCP	53	UDP	443	TCP	443	TCP	60001	TCP	443	TCP	53	UDP
443	TCP	389	UDP	53	UDP	80	TCP	80	TCP	0	IGMP	389	UDP
53	UDP	0	IGMP	49178	TCP	53	UDP	443	TCP	137	UDP	137	UDP
137	UDP	8080	TCP	49174	TCP	5353	UDP	137	UDP	60000	TCP	0	IGMP
0	IGMP	137	UDP	993	TCP	137	UDP	53	UDP	80	TCP	8080	TCP
445	TCP	8055	TCP	8443	TCP	5355	UDP	60000	TCP	445	TCP	80	TCP
60000	TCP	445	TCP	80	TCP	8443	TCP	445	TCP	57621	UDP	60000	TCP
389	TCP	80	TCP	137	UDP	993	TCP	0	IGMP	5222	TCP	443	TCP
9443	TCP	443	TCP	49181	TCP	5222	TCP	88	TCP	53	UDP	445	TCP
61845	TCP	138	UDP	49176	TCP	60000	TCP	389	TCP	49443	TCP	389	TCP
88	TCP	389	TCP	50102	TCP	61934	TCP	138	UDP	8080	TCP	8055	TCP
138	UDP	8014	TCP	50792	TCP	61795	TCP	8443	TCP	49503	TCP	61087	TCP
8443	TCP	88	TCP	5353	UDP	3389	TCP	389	TCP	8443	TCP	61084	TCP
61906	TCP	60000	TCP	445	TCP	61765	TCP	8014	TCP	389	TCP	138	UDP
8014	TCP	49155	TCP	55664	TCP	61682	TCP	49155	TCP	50708	TCP	88	TCP
389	UDP	135	TCP	55488	TCP	2598	TCP	58221	TCP	53323	TCP	49330	TCP
49155	TCP	5355	UDP	51227	TCP	49279	TCP	58226	TCP	49694	TCP	8443	TCP
61100	TCP	0	ICMP	53390	TCP	60117	TCP	135	TCP	56404	TCP	8014	TCP
135	TCP	9443	TCP	50103	TCP	61678	TCP	49793	TCP	49744	TCP	49197	TCP
8080	TCP	5353	UDP	53423	TCP	58725	TCP	0	ICMP	61348	TCP	49155	TCP

Masters Student		PhD Student		Program mgmt		Research Asst		Tenure		All Traffic	
Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot	Port	Prot
80	TCP	80	TCP	137	UDP	137	UDP	137	UDP	80	TCP
137	UDP	53	UDP	0	IGMP	0	IGMP	80	TCP	137	UDP
0	IGMP	443	TCP	3283	TCP	80	TCP	0	IGMP	0	IGMP
8080	TCP	5222	TCP	53	UDP	445	TCP	443	TCP	53	UDP
53	UDP	993	TCP	445	TCP	53	UDP	9443	TCP	8080	TCP
443	TCP	161	UDP	80	TCP	60000	TCP	8080	TCP	443	TCP
445	TCP	137	UDP	389	TCP	443	TCP	53	UDP	389	UDP
389	UDP	0	IGMP	138	UDP	389	TCP	445	TCP	445	TCP
9443	TCP	902	TCP	88	TCP	88	TCP	389	UDP	60001	TCP
8055	TCP	8443	TCP	8014	TCP	138	UDP	8055	TCP	9443	TCP
389	TCP	22	TCP	389	UDP	49334	TCP	389	TCP	60000	TCP
138	UDP	49170	TCP	443	TCP	65300	TCP	88	TCP	389	TCP
88	TCP	49168	TCP	49155	TCP	65286	TCP	8014	TCP	8055	TCP
8014	TCP	51496	TCP	135	TCP	8014	TCP	8443	TCP	138	UDP
60000	TCP	49167	TCP	5355	UDP	389	UDP	138	UDP	88	TCP
8443	TCP	51500	TCP	50497	TCP	49155	TCP	60000	TCP	8014	TCP
49159	TCP	49165	TCP	0	ICMP	135	TCP	49155	TCP	8443	TCP
135	TCP	51135	TCP	33355	UDP	8443	TCP	5355	UDP	135	TCP
49155	TCP	51145	TCP	771	ICMP	49359	TCP	135	TCP	49155	TCP
5355	UDP	52432	TCP	49351	TCP	3910	TCP	52217	TCP	49159	TCP

Table 4.3. Selected Ports and Protocols for Features

Port	Protocol	Often used for:
22	TCP	Secure shell
80	TCP	HTTP
88	TCP	Kerberos
137	UDP	NETBIOS Name Service
138	UDP	NETBIOS Datagram Service
389	UDP	LDAP
443	TCP	HTTPS
445	TCP	Microsoft Directory Services (SMB)
5222	TCP	Jabber/GoogleTalk Client Connection
5353	UDP	Multicast DNS
8080	TCP	HTTP-alt
8055	TCP	Senomix Timesheets Server
9443	TCP	VMware HTTPS, SSL
60000	TCP	MS Exchange RPC Client Access Service
60001	TCP	MS Exchange Address Book

($\overleftarrow{b_{p^2}}/|b|$), and the byte-value coefficient of variation (standard deviation normalized by the mean), or σ_{p^2}/μ_{p^2}. These features are referenced in Table 4.4 as port_X_in, port_X_out and port_X_std, respectively, where X refers to one of the port-protocol combinations listed in Table 4.3. In addition to these measures specific to the port-protocol listing in Table 4.3, we also created a feature measuring the entropy of the port-protocol counts observed during a measured interval. This set of port-protocol measures is referred to in Table 4.4 as port-behavior features, and are reused as a separate data set for comparing role-groups based on Netflow derived features.

Table 4.4 shows a listing of the statistical features extracted for our analysis, including features intended to summarize aspects of:

- Port behaviors
- Volume and flow density
- Protocol behaviors
- Handshaking (TCP flag) behaviors
- Temporal behaviors:
- IP address related measures

Table 4.4. Statistical and Information-Theory-Derived Features

Feature Name	Type	Description	Rationale
port_X_in	Port Behavior	Total port X bytes inbound/ total bytes all ports	Shows consumption of data passed for port service
port_X_out		Total port X bytes outbound/ total bytes all ports	Shows production of data passed for port service
port_X_std		Port X standard deviation/ mean of byte values	Shows uniformity of data passed for port service
port_entropy		entropy of distant ports	Diversity of services accessed
bytes_out	Volume & Flow Density	Total bytes outbound/total bytes passed	Ratio of data production/ data consumption
packets_out		Total packets outbound/total packets passed	Ratio of data production/ data consumption
bpp		Average bytes per packet	Density of traffic passed
tcp_frac	Protocol	TCP fraction of total flows	Reflects use of TCP based services
udp_frac		UDP fraction of total flows	Use of UDP based services
igmp_frac		IGMP fraction of total flows	Ue of multicast services
multicast		Fraction of multicast IP address flows (224.0.0.0/4)	Use of multicast services
flag_entropy	Handshaking	Entropy of TCP flag counts	Uniformity of TCP flag use
duration_std	Temporal Behavior	Standard deviation of flow duration values	Mix of short and long duration connections
interval_mean		Average interval between flow start times	Shows the density of flow occurrences during interval
interval_std		Standard deviation of flow start time intervals	Reflects mix of density of flow occurrences
ip_distance_ std	Address Related Features	Standard deviation of src/dst IP address distance/2^{32}	Reflects the diversity of address spaces connected to
ip_distance_ mean		Mean of src/dst IP address distance/2^{32}	Reflects mix of local and non-local connections
addr_entropy		Entropy of the IP addresses connected to	Measure of the diversity of address spaces connected to
direction		Fraction of flows outgoing	Ratio of data production/ data consumption

All combined, 61 different features were created. Most of the features generated were normalized based on an overall measure of the set of flows processed. Port_X_in measured the total inbound bytes passed over port-protocol X divided by the total bytes passed during the interval, giving the fraction of total bytes passed for that service. Likewise, bytes_out measured the fraction of total bytes that were in outbound flows, and tcp_frac measured the

fraction of flows using the TCP protocol. Measures such as bpp, flag entropy or ip_distance mean were not normalized, because they were not dependent on the total number of flows, packets or bytes passed during the interval.

Our decision to not use features reflecting absolute flow-volume measures was based on the observation that for some interval periods (in particular the longer periods), the active data transfers attributed to a user often spanned only a fraction of the interval. Using relative rather than absolute flow-volume features enables proportional comparisons between interval samples with unequal flow time spans.

4.1.2 Port Behavior Features

The second derived feature-vector-data set consisted of the port-behavior features described in Section 4.1.1. Port-behavior features are comprised of sets of individual measures (fraction of bytes in, fraction of bytes out, standard deviation of byte values) of the traffic for a given port-protocol combination, as well as a measure of the entropy of the different port-protocol combinations observed.

4.1.3 Port-Protocol Byte Value Distribution Features

Byte value distributions provide sequences of values describing the activity of a given port-protocol set of flows, with an additional dimension of behavioral description as compared to the individual measures described in Section 4.1.2. For each user data set and slicing interval, flow records were extracted for each port-protocol combination listed in Table 4.3, and flow byte value distributions computed for both outgoing and incoming flows. The distribution bins were roughly based on a logarithmic progression of byte value ranges: $0 > b \leq 41$, $41 > b \leq 80$, $80 > b \leq 160$, $160 > b \leq 320$, $320 > b \leq 640$, $640 > b \leq 1280$, $b > 1280$. The top value of the first bin range, $0 > b \leq 41$, was selected to count TCP packets with minimal payloads. The distributions were concatenated into one feature vector for each sampled interval.

4.1.4 Port Priority Vectors

Port Priority Vectors (PPVs) provide additional context in understanding an individual's usage of different port-protocol combinations, by providing a direct comparison with a

global (all user) norm (Section 3.2.2). The length of the PPVs generated for each sample interval was capped at 20 values. Index values for each PPV were capped at 500, as the higher index values in the global port-protocol reference list were primarily associated with ephemeral port values. If n port-protocol combinations are observed in the sample period and $n < 20$, the index values placed in vector positions $n + 1$ to 20 were set to 1000 to indicate incomplete lists.

4.1.5 Comparing Port-Behavior Representations

The feature vectors discussed in Section 4.1.2, Section 4.1.3 and Section 4.1.4 each represent different methods of expressing flow activity, with a focus on ports and protocols as a means of compartmentalizing the flow data. Applying these three different approaches to our Netflow data sets enabled a comparison of their utility in comparing the different role-group data sets. The statistical values are compact, leading to shorter feature vectors as compared to the distribution-based feature vectors. Distributions of byte data provide more detail in describing the types of flows that occurred (large transfers vs. small exchanges), but cause longer feature vectors. Our implementation of port priority vectors limited the vector lengths to 20 points, which made them the shortest of the feature-vector types tested.

4.2 Data Processing Factors

In performing our analyses, we created feature vectors derived from Netflow records extracted over defined intervals of time. To test the impact of the intervals used for extraction, we varied the intervals used to process both the unfiltered and cleaned Netflow records of each user.

4.2.1 Impact of Slicing Intervals

To convert flow data into feature vectors, for each user we divided their flow data into smaller chunks based on the times the flow activity occurred. If the start of the day in which data collection started is t_0 and the interval between slicing period starts is Δt, then for every interval $t_0 + n\Delta t$ to $t_0 + (n + 1)\Delta t$ during which flow data was recorded, for each user their flows during that interval were used to create feature vectors. A flow starting in one slicing period and ending in another would be split proportionally, such that each

spanned interval would include a flow record with the same five-tuple (source IP address, destination IP address, source port, destination port and protocol) but with packet and byte values proportional to the fraction of flow duration that overlapped that interval.

For each user ID assigned to a role-based group, all flow records (across one or more hosts, during one or more periods) associated with that user were combined, sorted chronologically by flow start times and divided based on a selected interval value. Interval samples in which flow activity covered only a small fraction of the interval period (time between first flow start and last flow end < 0.1 x interval period) were discarded.

During each slicing interval, flow records associated with the user (if present) were processed to create feature-value vectors, i.e. arrays of values in which each position in the array corresponds to either a feature (see Table 4.4) or a position in a byte value distribution or list (see Section 4.1.3). Each feature vector provides a representation of the flow records generated by a user during one of evenly separated intervals of network activity, by containing features, distributions or lists that measure specific aspects of the flow activity during the interval.

We computed our feature vectors over multiple intervals (15, 30, 60 minutes and one day), to test which slicing interval resulted in the best association of the feature vectors with user roles. Data slicing over shorter intervals allows capturing more transient user behaviors, which can be expressed as greater variability in terms of feature values. Longer slicing periods capture longer term summaries of behavior, in which the feature value variations caused by transient activities are averaged out. Each vector was labeled with the role group (Table 3.3) the user was assigned to.

4.2.2 Impact of Data Cleaning

For each slicing interval, feature vectors of each type (baseline, port-behavior, port distributions and PPVs) were created twice, once based on flow record data sets before data cleaning and again after cleaning. Cleaned flow records were flow data sets in which the flows tagged as being automatically generated were removed. Both cleaned and unfiltered feature-vector- data sets were tested in our experiments, to observe whether cleaning the data improved the abilities of the classifiers to differentiate between role-group-data sets.

4.3 Data Pre-Processing

For each of the data classification experiments, the feature set data was pre-processed to:

- Down-sample the larger role-group data sets to reduce class imbalances
- Normalize the values for each feature in the feature value set to zero mean and unit variance
- Reduce the effective feature vector lengths through Principal Component Analysis (PCA), for the longer (non-PPV) feature-vector types.
- Create a pseudo role-group, to serve as a control group for the experiments

4.3.1 Down-Sampling of Larger Role-Group Data Sets

The membership for the role-groups listed in Table 3.3 is unbalanced, and as a result the number of flow data samples extracted for each role group is also unbalanced. The Tenure and Masters Students role-groups were associated with the largest numbers of extracted feature vectors. To prevent these groups from completely dominating classifier decisions, for each iteration of data classification the data from the Tenure group was randomly down-sampled by 50%, and data from the Masters student group was down-sampled by 85%. Even with the down-sampling of the largest groups, the data was still unbalanced as shown in Figure 4.1, which shows the number of per role-group vectors after down-sampling for both cleaned and unfiltered flow data sampled at 15-minute intervals.

4.3.2 Data Normalization

For each iteration of the experiments, the values for each feature in the selected data set were normalized to have a zero mean and unit variance. This was done to ensure each feature in the extracted feature vectors contribute equally in classifying the data.

4.3.3 Dimension-Reduction of Feature-Vectors

As the number of features in a feature vector increases, the dimensionality of the space that distance measurements between vectors are made within also increases. Increasing dimensionality reduces the impact that any one feature has on overall distances between vectors. This effect is known as the "Curse of Dimensionality" [64]. The feature-vector lengths obtained by concatenating different feature sets can create very long vectors. To

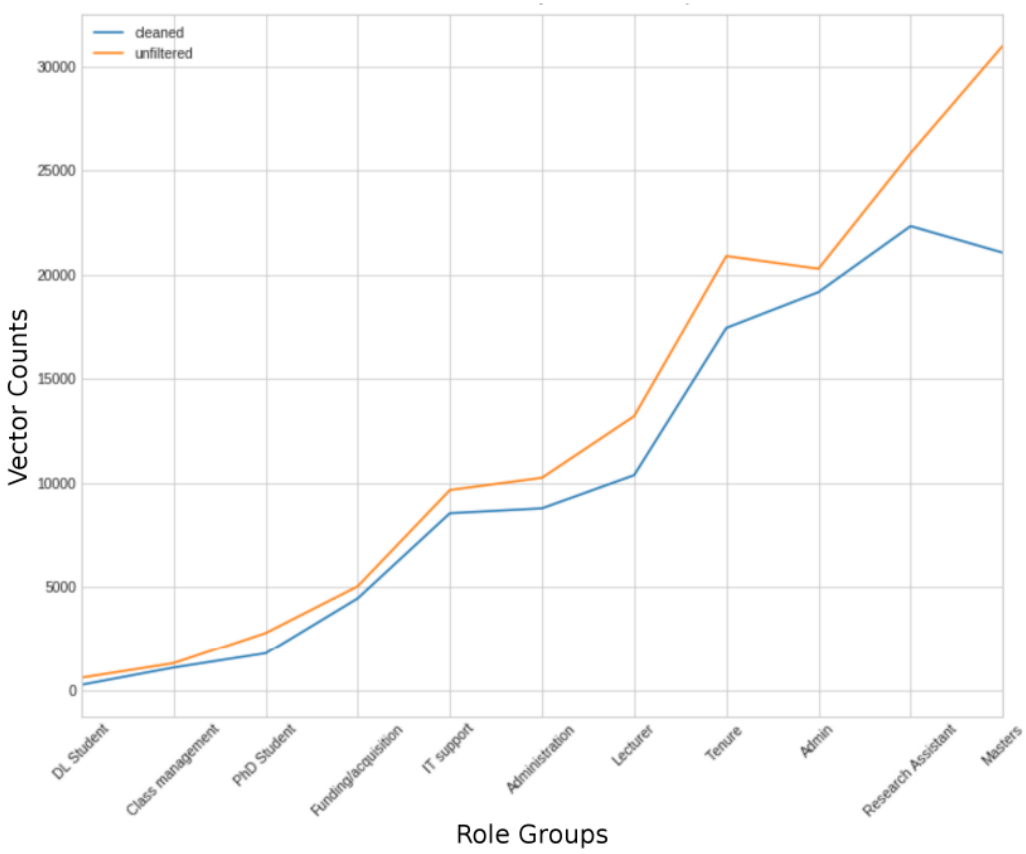


Figure 4.1. Vectors per Role Group

address this, we applied Principal Component Analysis (PCA) to each feature vector set, reducing the number of dimensions to a number that conserved 95% of the variability found in the data sets. Using this approach, the statistically-derived, port-behavior and port-distribution feature-vector dimensions were reduced to 31, 52 and 28 dimensions respectively.

4.3.4 Pseudo-group Creation

To test the relationship between a user's role and the flow data they generate, we created control (pseudo-role) groups designed to be non-role specific. The pseudo-role groups were generated by randomly selecting users from each of the defined role groups, and extracting without replacement the data vectors associated with them. Because the number of feature

vectors per user varied, as each user was selected the total number of associated feature vectors was counted. If the total was between 13 and 17 percent of the total vectors for the role group, random selection and extraction for that role group was stopped. If adding a user associated with a large number of feature vectors caused the total selected feature-vector count to exceed 20 percent, the data from that user was not extracted.

The intent of creating the pseudo-role group was to determine whether classifiers would perform any differently with data from an arbitrarily defined set of users than they would with data extracted from our defined role groups. If the performance of the classifiers with pseudo-group data is approximately the same as with our defined role groups, then the use of roles in creating our groups of users did not enhance our analysis of user behaviors.

4.4 Role-Based User Group Experiments

4.4.1 Data Classification

The classification experiments performed to detect and measure the relationship between user roles and the network traffic the users generated contained several dimensions of investigation. These dimensions were:

- Use of two different classifiers, one linear (Nearest Centroid Classifier) and the other non-linear (Support Vector Machine with a radial basis function kernel)
- Use of four different feature sets (Section 4.1)
- Use of unfiltered and cleaned Netflow records
- Use of four data slicing intervals (15 minutes, 30 minutes, 60 minutes and 1 day)

For each combination of these dimensions (classifier, feature set, data cleaning and slicing period), classification experiments were repeated 10 times. For each iteration the set feature vectors were pre-processed (Section 4.3). The classifier was trained on a subset of the feature vectors (70% randomly selected from each of the role-groups) and tested on the remaining 30%. Classification results from the 10 iterations were averaged. The classification experiments were performed twice, once for each version of pseudo-group generation.

4.4.2 Data Clustering

Feature vectors derived from flow records that summarize a user's data transfers over the network provide some measure of the flow patterns generated by that user. These patterns can be considered to be a description of user behaviors over the network, and if user roles strongly influence user behaviors we would expect that users in the same role group would share similar behaviors on the network.

Clustering algorithms associate data points based on relative distances; data points can be grouped as belonging to a cluster if the distances between the points are small relative to distances to other data points. If user network behaviors are in part determined by their organizational role, we would expect that users in the same role group would share common behaviors which would be reflected in sharing similar feature vectors in their data sets. Clustering the data sets of all the users should lead to the generation of clusters containing feature vectors derived from one or two of the role groups.

For each of the four feature sets (Section 4.1), data vectors were clustered using k-means++ to determine if the clusters formed reflected the similarities within each role-group.

4.4.3 Feature-Vector Distance

While classification algorithms can be used to test how separable feature data is for a few classes, they are less useful when comparing data sets associated with hundreds of classes. To compare the differences between users within the role-groups, we performed a pairwise comparison between the users in our role groups based on the feature vectors associated with each user.

Let $U = \{u_1, ..., u_i, ..., u_m\}$ be the set of users identified for our research, and $V_i = \{v_1, ..., v_k, ..., v_n\}$ the set of feature vectors associated with a user u_i. For each $u_i \in U$ and $u_j \in U$, let d_{ij} represent the mean of the pairwise euclidean distances between the feature vectors in V_i and V_j. For m users, this produces an $m \times m$ matrix of distance values, D.

Let $G = \{g_1, ..., g_k, ..., g_p\}$, where g_k is one of the defined role groups each containing a subset of the users. For each subset of users in each $g_k \in G$, we collected three types of distance-value distributions. The first distribution focused on the self-similarity of feature

vectors produced by the same user. For each user u_i in group g_k we collected the d_{ii} distance values, or the mean-pairwise distance between the feature vectors in v_i. This set of self-similarity distances provide a measure of how similar feature vectors generated by the same user are over time.

The second distance distribution focused on the mean distances between the feature vectors of pairs of users in group g_k. For each user $u_i, u_j \in g_k$ where $i \neq j$, we collected the d_{ij} distances. This set of distances show how similar feature vectors generated by users in the same role group are.

The third type of distance distributions focused on the mean pairwise distances between feature vectors generated by users in group g_k and the feature vectors generated by users in other role groups. For each role group g_l, where $l \neq k$, we collected mean pairwise distances d_{ij} between each user $u_i \in g_k$ and user $u_j \in g_l$. This set of distances show how similar the feature vectors generated by users in one role group are to feature vectors generated by users in other role groups.

4.5 Similarity-Based User Group Experiments

The underlying assumption behind grouping users by roles in order to define normal user network activity is that users within the same role group would perform similar tasks, and so the patterns of network traffic for these users would be similar. If true, feature-value distributions derived from the network traffic of users in the same role group could be used to define the bounds of normal behavior for users in that role-group. Another approach to identifying users with similar network behaviors is to not assume similarities based on roles, but to observe their network behaviors and group them by similarity.

To find groups based on similarities in user behaviors, we adapted some of the methodology described by Frias-Martinez [3] (see Section 2.3.2). Frias-Martinez clustered feature vectors derived from user-flow-data sets to identify users with similar behaviors, and compared new feature vectors for each user against the existing clusters. Feature vector distances too far from the majority of points in a user's cluster were declared anomalous. Our process was not as complex as the methods used by Frias-Martinez, but it served to demonstrate that user-data sets can be clustered based on similar behaviors.

For a given set of feature vectors, the vectors are grouped based on the week the represented data was collected. For each user and week, centroid vectors (vectors of mean values for each feature in a feature-vector set) are calculated. The centroid vectors are clustered using K-means++, and user groups defined by which user centroids grouped in each cluster. A pseudo group is generated by extracting users and associated data vectors from the defined groups to test whether classifiers performed more poorly with a mixed-group set.

Using the redefined groups, the experiments in Section 4.4 are repeated, to determine if the relabeled-user groups present different results as compared to role-based groups when tested for similarity via classifiers, clustering, and user-data-set distance comparisons.

CHAPTER 5:
Results and Discussion

This chapter provides the results of the experiments discussed in Chapter 4, and analyses of the experimental results. The suitability and limitations of employing single features for comparing user network behaviors is reviewed in Section 5.1. The performance of the nearest centroid and SVM classifiers using vectors of statistical and information theory based features is discussed in Section 5.2. These features include volume-based port-behavior measures (bytes in/total bytes passed, bytes out/total bytes passed, standard deviation of byte values) over the ports and protocols listed in Table 4.2, as well as statistical and information-theoretic measures based on flow protocol, TCP handshaking, flow temporal behavior, and IP address values (listed in Table 4.4). This feature set provides a baseline for classifier performance relative to the other data sets, in that it incorporates both port-protocol flow volume measures as well as the other statistical and information-theoretic features described in Table 4.4.

In the following three sections, we discuss the performance of the classifiers against different feature-vector types used to describe flow patterns over the different ports and protocols. In Section 5.3, we discuss the performance of the classifiers on a subset (Port-Behavior) of the features listed in Table 4.4. These features consist of measures of flow activity over the port-protocols listed in Table 4.3, plus a measure of distant-port entropy. In Section 5.4 the performance of the classifiers on feature vectors consisting of flow byte-value distributions for each of the selected ports and protocols is discussed. The performance of the classifiers using Port Priority Vectors as the discriminating features is covered in Section 5.5. Section 5.6 discusses the impact of consolidating the 11 user-role groups into three more general user categories and equalizing data set sizes prior to classification testing, and comparison of classification results for all of the feature-vector types is discussed in Section 5.7.

The results observed from clustering the different feature vector types using K-means++ are reviewed in Section 5.8, and Section 5.9 discusses the relative feature-vector distances between users; to themselves, to others in their role group and to users in the other role

groups. Finally, Section 5.10 discusses the results obtained from grouping user data sets based on behavioral similarities, and repeating the tests performed against the role-based user groups.

5.1 Single Feature Discriminators

We will not describe here the many negative results we obtained with simpler features on this data. For instance, as can be seen in Figure 5.1, total bytes in the flow is not a good discriminator of user groups since flows vary too much in this measure and this overrides any effect of user group. The same can be said for measuring the bytes per packet in flows (Figure 5.2).

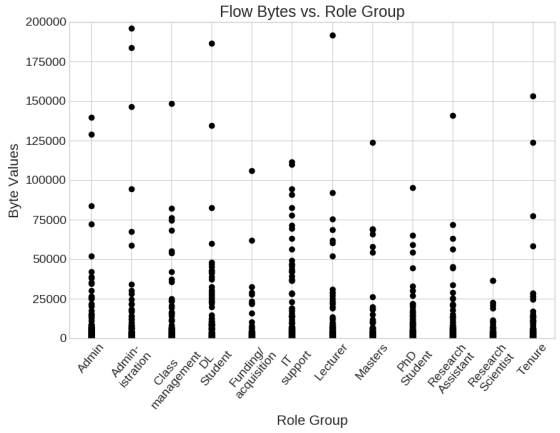


Figure 5.1. Flow Bytes vs. Role Group

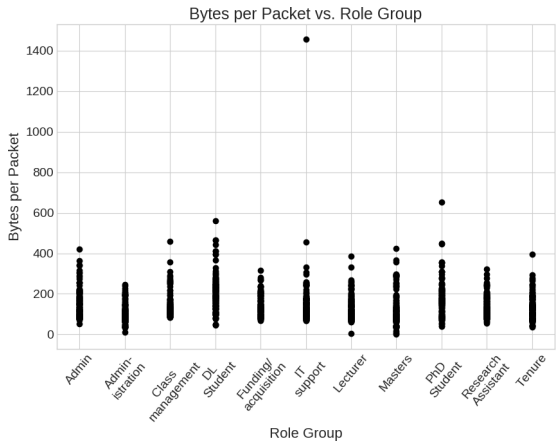


Figure 5.2. Bytes Per Packet vs. Role Group

After such tests, we concluded that most features are not useful on an individual basis for discrimination between user behaviors. But use of features together provides more detail for comparison.

5.2 Aggregate Netflow Statistics

We tested our baseline set (Table 4.4) of features using two different machine learning algorithms, a nearest-centroid classifier and a support vector machine.

5.2.1 Nearest Centroid Classification

To perform these experiments, we used the nearest centroid classifier provided by the scikit-learn python library [63]. The classifier was trained and tested on value vectors containing the features described in Table 4.4. Feature-vector data sets were created eight times, for each combination of data cleaning (cleaned and not cleaned) and data slicing interval (15, 30, 60 minute and one day).

Classification trials for each feature-vector-data set were repeated 10 times, and the results averaged. Averaging of the results was necessary due to the random sampling applied for each iteration to:

- Down-sample the larger (Tenure and Masters Student) group data Section 4.3.1,
- Extract between 13-17% of feature vectors from each role group to create the pseudo group data set Section 4.3.4, and
- Extract 30% of the feature vectors from each role group to create the test data set.

Sampling was performed through feature vector extraction without replacement. For each role-group classifier, the number of feature vectors selected from each of the role-groups was averaged across the 10 trials. These average values were used to create the confusion matrices and plots presented in this section.

Confusion Matrix Analysis: Table 5.1 shows the confusion matrix for the nearest-centroid classifier, when trained and tested on features described in Table 4.4. The data used to create the feature vectors used for this experiment was derived from non-cleaned flow data (no removal of flows flagged as automatically generated), that was sliced on 30-minute intervals.

For each cell $c_{i,j}$ in the confusion matrix, the cell shows the number of feature vectors (averaged across the 10 trials) from role-group i labeled by the classifier as belonging to role-group j. True positive classifications are counted in those cells where the column and row role-group names match. In Table 5.1 the matrix cells with the true positive values for each role-group classifier are bounded by border lines, while each cell $c_{i,j}$ containing bolded numbers indicate the role-group i with the maximum number of feature vectors selected by the classifier as belonging to role-group j. The precision and recall measures for each role group are provided in the corresponding labeled row and column bordering the confusion

Table 5.1. Non-Cleaned Data Confusion Matrix

	Admin	Administration	Class Mgmt	DL Student	Funding/acq	IT Support	Lecturer	Masters Student	PhD Student	Research Asst	Tenure	Pseudo	Recall
Admin	**1140**	12	183	118	**448**	289	39	9	23	101	2	197	0.45
Administration	323	**566**	33	53	115	49	32	66	24	72	8	81	0.40
Class Mgmt	9	0	112	10	10	13	0	0	2	11	0	4	0.65
DL Student	5	0	5	80	0	0	1	0	12	6	0	2	0.73
Funding/acq	234	0	58	64	157	93	9	9	15	33	0	21	0.23
IT support	340	53	106	77	207	409	13	18	19	64	3	37	0.30
Lecturer	560	231	129	173	274	85	71	46	44	92	21	66	0.04
Masters Student	685	543	144	481	263	287	**75**	**128**	113	163	27	147	0.04
PhD Student	3	0	64	142	2	9	0	0	**157**	17	0	32	0.37
Research Asst	925	1	**374**	291	432	**421**	28	19	108	**701**	20	287	0.19
Tenure	582	503	168	326	287	239	61	68	102	234	**61**	297	0.02
pseudo	361	215	229	**505**	184	303	36	41	103	209	11	**337**	0.13
Precision	0.22	0.27	0.07	0.03	0.07	0.19	0.20	0.32	0.22	0.41	0.40	0.22	

matrix.

Of the 12 role groups shown in Table 5.1, seven of the true-positive values were also the maximum selected (shown in bold) for their column's role group (including the pseudo group). In other words, in those cases the classifier selected more feature vectors from the correct role group than from any of the other role groups. For the pseudo group the classifier had a precision of 0.22 and a recall of 0.13, performing as well as or better than the scores received for some of the other role groups.

The confusion matrix shown in Table 5.2 shows the results of training and testing the nearest-centroid classifier using feature vectors derived from data sliced over 30-minute intervals and cleaned of flows flagged as automatic. Removing the automatic flows did not change the results significantly; the classifier again recalled more feature vectors from the correct role group than selected from the other groups for seven of the 12 role groups. Classification of the pseudo-group data had a precision score of 0.06 and a recall score of 0.33, not too dissimilar to the performances for the Tenure and Admin role groups.

Table 5.1 and Table 5.2 provide detailed results on the testing results for the nearest-centroid

Table 5.2. Cleaned Data Confusion Matrix

	Admin	Administration	Class Mgmt	DL Student	Funding/acq	IT Support	Lecturer	Masters Student	PhD Student	Research Asst	Tenure	Pseudo	Recall
Admin	**195**	374	338	237	**462**	381	**190**	52	51	190	6	41	0.08
Administration	28	382	46	117	130	78	101	178	32	87	24	21	0.31
Class Mgmt	4	0	116	11	11	12	0	0	4	7	0	1	0.69
DL Student	2	0	0	37	0	0	0	0	4	0	0	0	0.86
Funding/acq	9	90	71	35	195	89	58	19	20	44	1	6	0.31
IT support	79	65	145	42	248	**432**	45	33	28	60	6	20	0.36
Lecturer	39	334	130	116	279	166	176	103	66	58	19	25	0.12
Masters Student	73	501	213	150	215	300	142	**348**	78	61	37	43	0.16
PhD Student	5	0	72	74	1	4	0	0	**121**	12	3	5	0.41
Research Asst	127	326	**416**	191	457	411	178	94	105	**730**	23	61	0.23
Tenure	103	**437**	233	222	317	268	185	264	108	166	**136**	67	0.05
pseudo	105	412	232	**240**	304	209	178	232	76	123	29	**144**	0.06
Precision	0.25	0.13	0.06	0.03	0.07	0.18	0.14	0.26	0.17	0.47	0.48	0.33	

classifier on two of the eight versions ([clean vs. not cleaned] x [four slicing intervals]) of derived data sets. For the rest of this chapter, we will primarily display test results in graph form, which is more succinct.

Graphical Analysis: Figure 5.3 shows the nearest-centroid classifier's precision and recall scores for each slicing interval value on both the unprocessed and cleaned Netflow data sets. We ordered the sequence of role-group names listed on the independent variable (x) axis based on the sizes of the role-group feature-vector-data sets, largest data set first and the other groups listed in descending set-size order. This ordering of group names enables easier comparisons of the precision and recall value relationships, as well as the relationships of the scores to the role-group-set sizes. Note: the relative sizes of the role-group data sets were determined after the extraction of user-feature vectors to create the pseudo-role group.

Several observations can be made based on Figure 5.3:

- With some exceptions, average precision scores for each role group decreased as the number of feature vectors per role group decreased. The correlation was not exact; for example the Admin-role group received lower precision scores than observed for

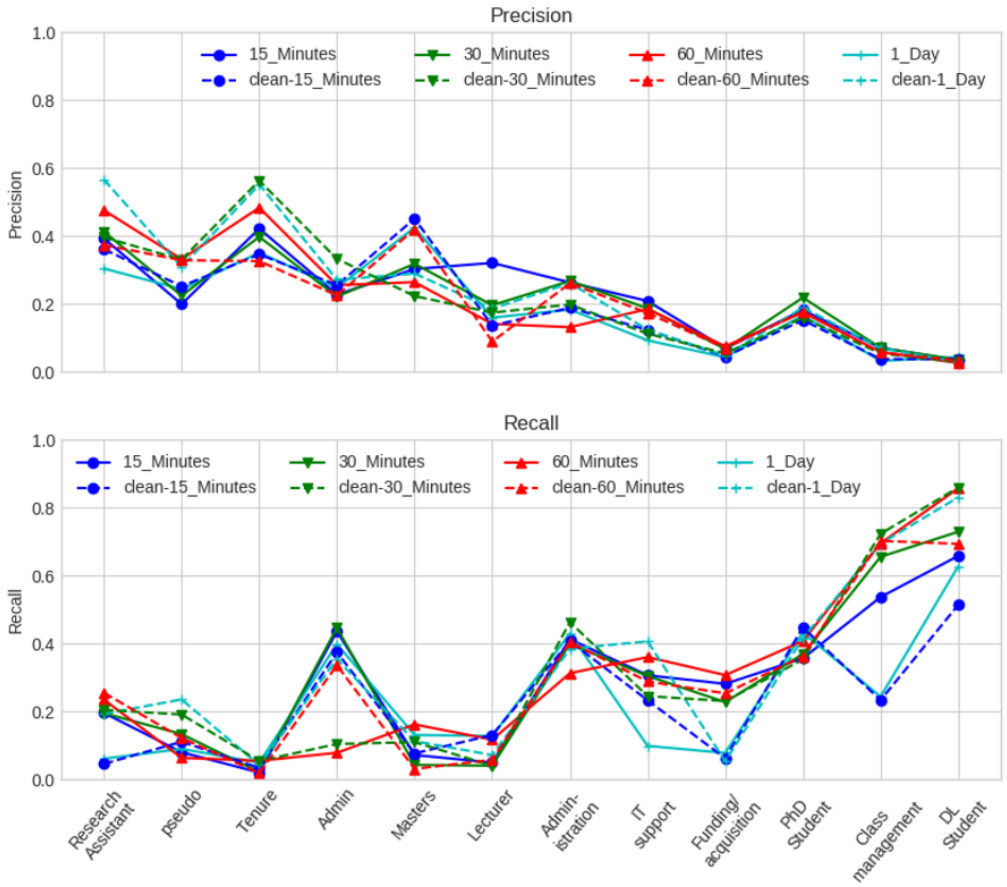


Figure 5.3. Baseline Set Precision/Recall Scores for Nearest Centroid Classifier

the smaller Masters-role group.

- Cleaning the data of automatic flows did not appear to create a consistent difference in the precision or recall measures relative to flow data that was not cleaned.
- The pseudo group had precision and recall scores comparable to those of the Research Assistant and Tenure-role groups, which were similar in feature-vector-set size.

Based on the results shown in Figure 5.3, there appears to be a strong correlation between the size of the role-group data sets and the precision scores achieved by the classifier. To test this apparent correlation, we set the maximum role-group vector-set size to be no larger than that of the IT Support role group. Larger data sets were randomly sampled for each classifier test iteration, selecting a number of feature vectors equal to that of the IT Support

group. Figure 5.4 shows the classifier results of the down-sampled data.

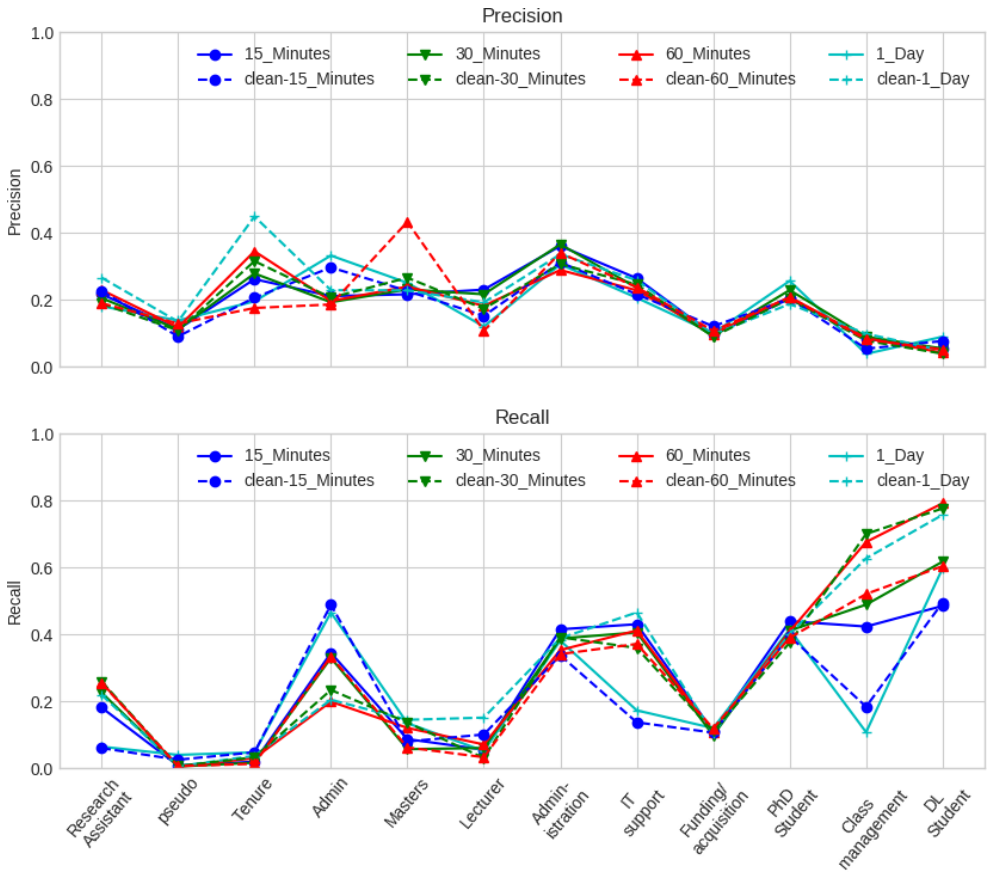


Figure 5.4. Down-sampled Baseline Set Precision/Recall Scores for Nearest Centroid Classifier

As can be seen in Figure 5.4, down-sampling the larger groups appears had an impact on the classification results. The average precision results for the larger role-groups flattened out for the larger data sets. The recall values had dropped significantly, however. Because down-sampling did not significantly improve the overall classification patterns observed, only the largest role-group data sets (Masters and Tenure) were down-sampled (as described in Section 4.3.1) for the remaining classification tests.

5.2.2 Support Vector Machine Classification

Testing and training of the Support Vector Machine classifier was performed in the same manner and on the same data as the nearest-centroid classifier tests. To perform these

experiments, we used the support-vector-machine classifier provided by the scikit-learn python library [63]. As was done for the nearest-centroid classifier, the classification trials were repeated 10 times and the average results collected.

Confusion Matrix Analysis: Table 5.3 shows the confusion matrix for the Support Vector Machine classifier tested and trained on the features described in Table 4.4, extracted from user flow data that was sliced on 30-minute intervals and not cleaned of automatic flow records. Table 5.4 shows the confusion matrix for the SVM trained on the same features extracted from cleaned flow data, again sliced on 30-minute intervals.

Table 5.3. Non-Cleaned Data SVM Confusion Matrix

	Admin	Administration	Class Mgmt	DL Student	Funding/acq	IT Support	Lecturer	Masters Student	PhD Student	Research Asst	Tenure	Pseudo	Recall
Admin	**1000**	25	**239**	89	**748**	176	9	37	154	118	20	50	0.38
Administration	132	**682**	34	91	128	31	44	54	76	25	23	26	0.51
Class Mgmt	17	0	125	2	8	1	0	0	6	3	1	3	0.75
DL Student	0	0	1	94	0	0	3	1	9	1	1	2	0.84
Funding/acq	111	2	11	40	393	56	3	10	29	20	4	9	0.57
IT support	138	69	18	98	249	**475**	10	111	35	43	14	26	0.37
Lecturer	167	351	31	180	437	135	**180**	147	96	49	34	37	0.10
Masters Student	141	623	64	**586**	102	172	77	**909**	162	69	71	87	0.30
PhD Student	10	4	7	75	7	5	7	7	**280**	13	8	5	0.65
Research Asst	455	21	172	258	640	400	18	153	252	**1100**	37	66	0.31
Tenure	348	572	118	261	394	169	67	182	180	95	**443**	74	0.15
pseudo	323	321	131	297	191	142	41	220	117	80	52	**656**	0.26
Precision	0.35	0.26	0.13	0.05	0.12	0.27	0.39	0.50	0.20	0.68	0.63	0.63	

As can be observed in Table 5.3 and Table 5.4, the SVM classifier showed better recall performance than the nearest-centroid classifier. For the non-cleaned data, the SVM selected more of the correct role-group data vectors for 9 of the 12 role groups. For the cleaned data, the SVM selected more correct role-group vectors for 8 of the 12 role groups. As was observed for the nearest-centroid classifier the precision and recall scores for the pseudo-group detection in the two tables were not distinctly different from some of the scores achieved for the original role groups.

Table 5.4. Cleaned Data SVM Confusion Matrix

	Admin	Administration	Class Mgmt	DL Student	Funding/acq	IT Support	Lecturer	Masters Student	PhD Student	Research Asst	Tenure	Pseudo	Recall
Admin	**1220**	3	**214**	22	**463**	171	47	125	92	138	26	52	0.47
Administration	141	**605**	32	29	66	30	32	104	38	30	38	23	0.52
Class Mgmt	19	0	124	1	4	1	2	7	2	4	1	3	0.74
DL Student	0	1	2	32	1	0	2	2	2	0	1	2	0.73
Funding/acq	168	2	13	2	298	29	10	22	17	18	6	5	0.51
IT support	176	4	27	17	171	**610**	14	49	10	48	8	14	0.53
Lecturer	276	168	21	36	194	134	**269**	190	55	60	37	23	0.18
Masters Student	126	387	65	33	75	180	63	**931**	50	90	87	72	0.43
PhD Student	10	2	8	21	5	4	7	16	**204**	9	5	4	0.69
Research Asst	559	15	152	18	388	457	52	219	186	998	25	32	0.32
Tenure	355	465	104	34	159	236	69	281	110	103	**520**	59	0.21
pseudo	449	396	135	**51**	182	202	49	349	85	93	66	**415**	0.17
Precision	0.35	0.30	0.14	0.11	0.15	0.30	0.44	0.41	0.24	0.63	0.63	0.59	

Graphical Analysis: Figure 5.5 shows the SVM classifier precision and recall scores for each slicing interval value on both the unprocessed and cleaned Netflow data sets. Several observations can be made based on Figure 5.5:

- As was seen in Section 5.2.1, decreasing precision measures roughly correlated with the decreasing numbers of feature vectors per role group.
- Both precision and recall measures based on the one-day slicing interval were more often the minimum or maximum values per role group. Measures based on the 15, 30 or 60-minute slicing intervals tended to be closer in value.
- Cleaning the data of automatic flows did not appear to create a consistent difference in the precision or recall measures.
- The pseudo group had precision and recall values between those of the Research Assistant and Tenure role-groups, which were similar in terms of feature-vector set sizes.

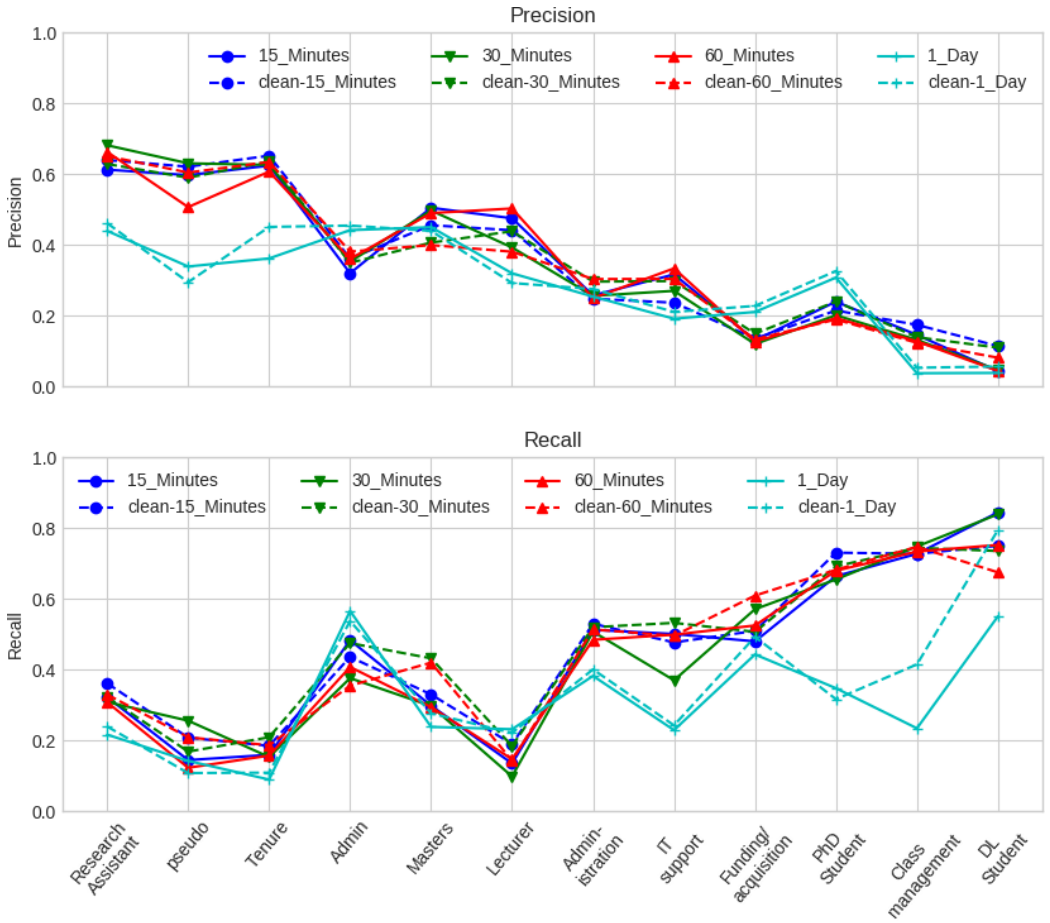


Figure 5.5. Baseline Set Precision/Recall Scores for SVM Classifier

5.3 Port Volumetric Feature Analysis

The data vectors used for this analysis consisted of the port behaviors features listed in Table 4.4. As described in Section 5.2.1, the Netflow data was processed for each combination of data cleaning (cleaned and not cleaned) and data slicing interval (15, 30, 60 minute and one day). Classification trials were repeated 10 times, and the averaged results used to create the plots in this section.

5.3.1 Nearest Centroid Classification

Figure 5.6 shows the nearest-centroid classifier precision and recall scores for each of the port-behavior feature-vector data sets. As was done in the result graphs in Section 5.2, the sequence of role-group names listed on the independent variable (x) axis was ordered based

on the sizes of the role-group feature-vector-data sets, largest data set first and the other groups listed in descending set-size order. The precision and recall results were lower than what was observed in Section 5.2.1, implying that the non-port-behavior features included in Table 4.4 did contribute to the classifier's performance. Average pseudo group precision and recall values were similar to those of the Research Assistant and Tenure-role groups.

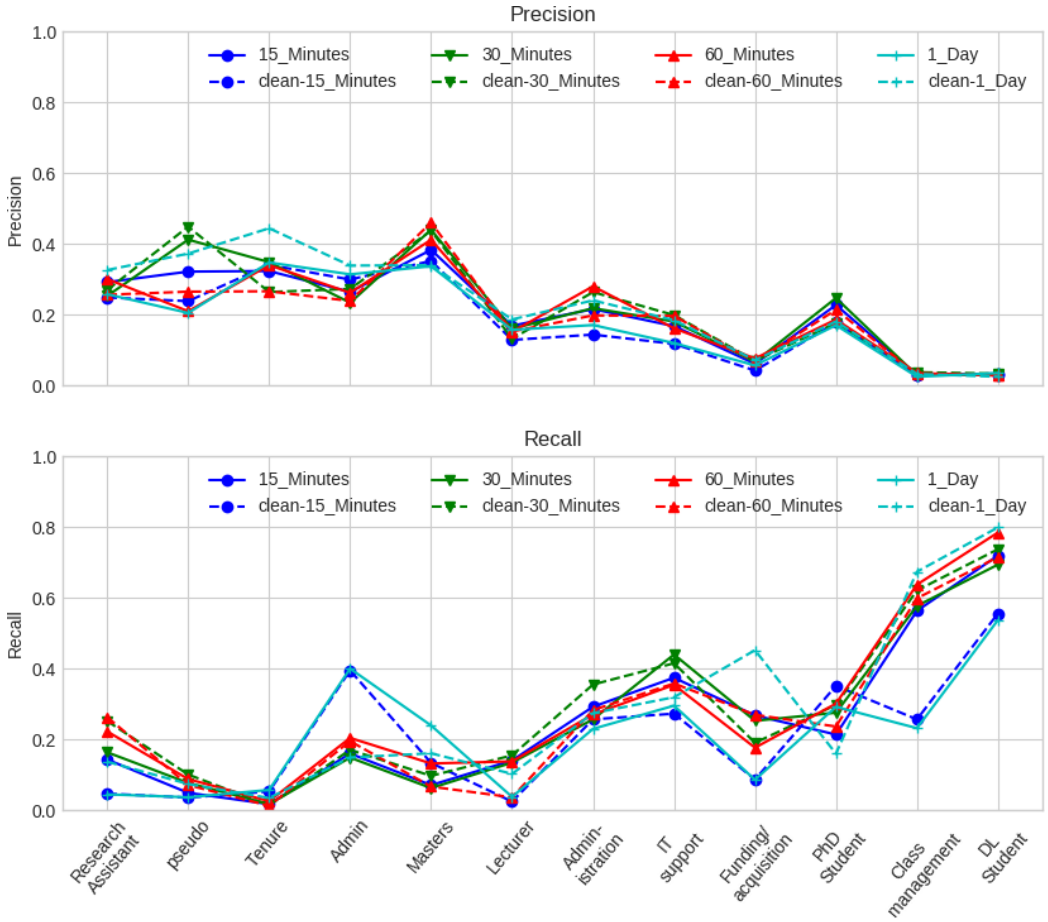


Figure 5.6. Port Volumetric Set Precision/Recall Scores for Nearest Centroid Classifier

5.3.2 Support Vector Machine Classification

Figure 5.7 shows the precision and recall scores for the SVM classifier using the port-behavior based feature-vectors. The general trends observed in Section 5.2.2 apply for the results for this feature set as well. Contrary to what was noted in Section 5.3.1, the

precision and recall values for the port-behavior only classification results were similar to those achieved when using the all features listed in Table 4.4.

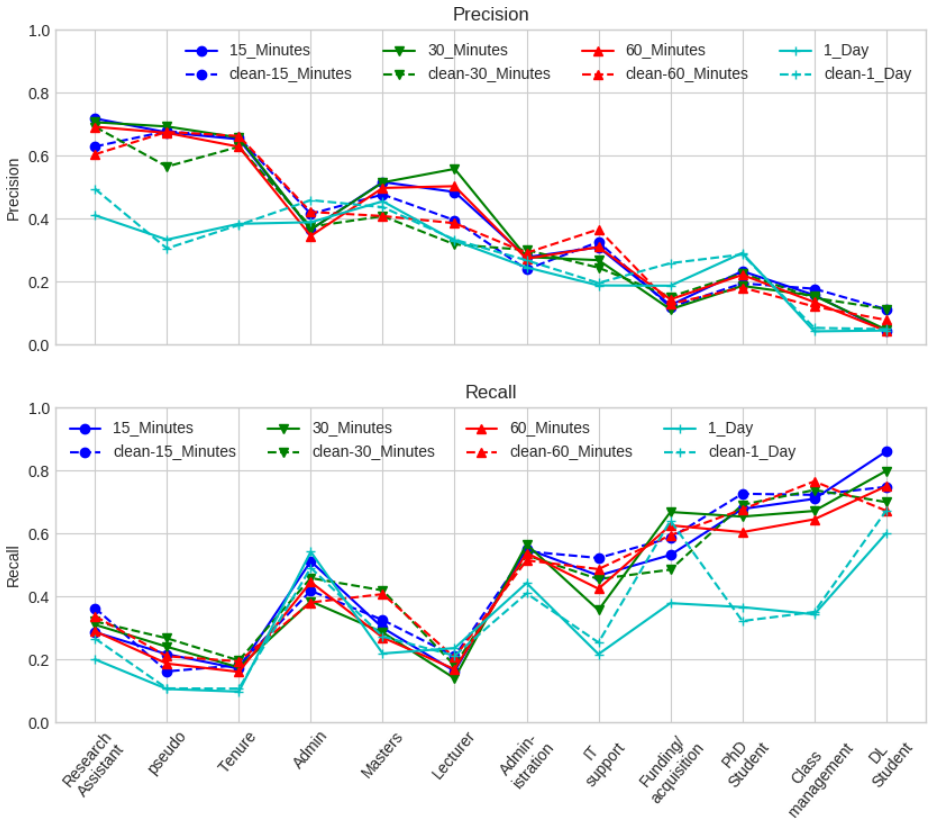


Figure 5.7. Port Volumetric Set Precision/Recall Scores for SVM

5.4 Port Distributions Analysis

The data vectors used for this analysis consisted of byte-value distributions (Section 4.1.3) for both incoming and outgoing traffic over the port-protocol combinations listed in Table 4.3. As described in Section 5.2.1, the Netflow data was processed for each combination of data cleaning (cleaned and not cleaned) and data slicing interval (15, 30, 60 minute and one day). Classification trials were repeated 10 times, and the averaged results used to create the plots in this section.

5.4.1 Nearest Centroid Classification

As was performed with baseline feature set discussed in Section 5.2, the sequence of role-group names listed on the independent variable (x) axis is ordered based on the sizes of the role-group feature-vector-data sets, largest data set first and the other groups listed in descending set-size order. Analysis results for training and testing the nearest-centroid classifier on the byte-value distribution vectors is shown in Figure 5.8.

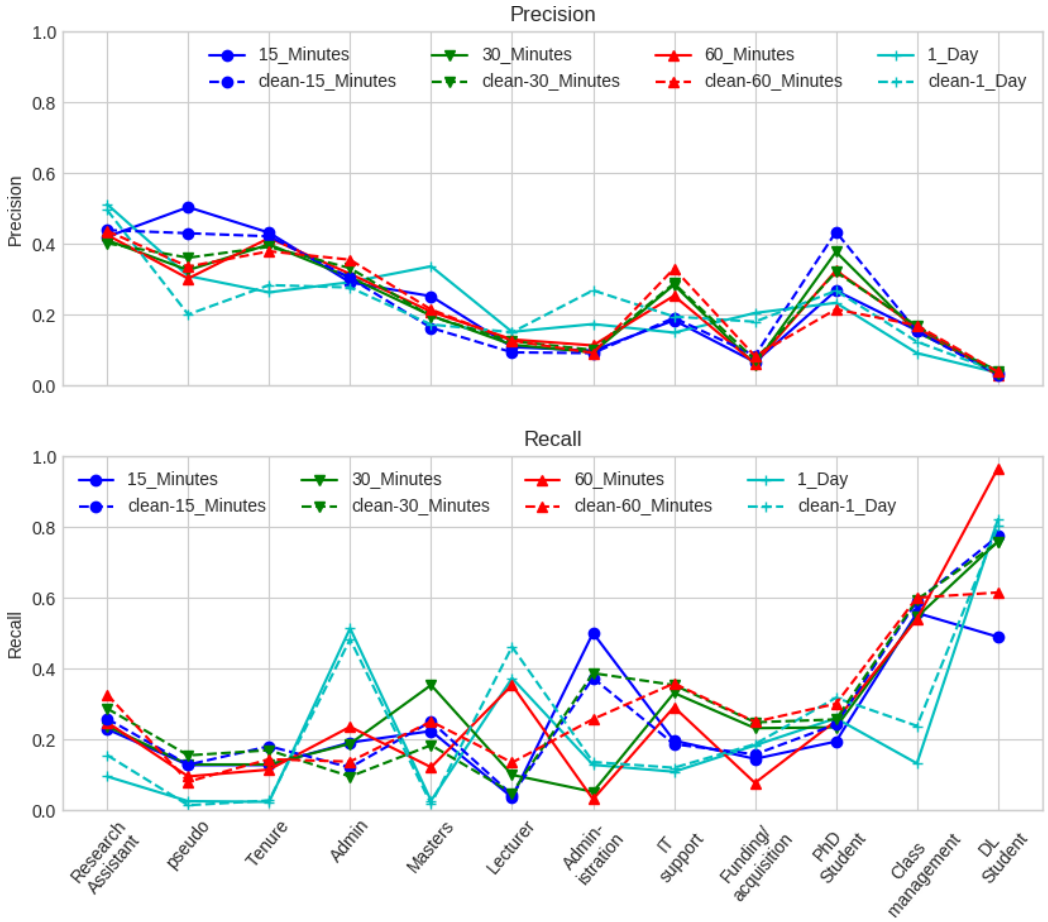


Figure 5.8. Port Distribution Set Precision/Recall Scores for Nearest Centroid Classifier

Several observations can be made based on Figure 5.8:

- As observed in Figure 5.3 and Figure 5.6, cleaning the data had no consistent effect on the precision and recall values. The average pseudo-group recall and precision values were similar to those received by the Research Associate and Tenure-role groups.

- The group-size, precision-value relationships observed in Section 5.2 and Section 5.3 were more mixed for the port-protocol byte distribution data vectors, in that the trend in precision values did not correlate as closely with the role-group data-sample size.
- The precision and recall measures based on the one day slicing interval deviated significantly from the per-role average values of the other slicing intervals. Measures based on the 15, 30 or 60-minute slicing intervals tended to be closer in value.

5.4.2 Support Vector Machine Classification

Figure 5.9 shows the precision and recall scores for the SVM classifier using the port-protocol byte-value distribution-based feature-vectors.

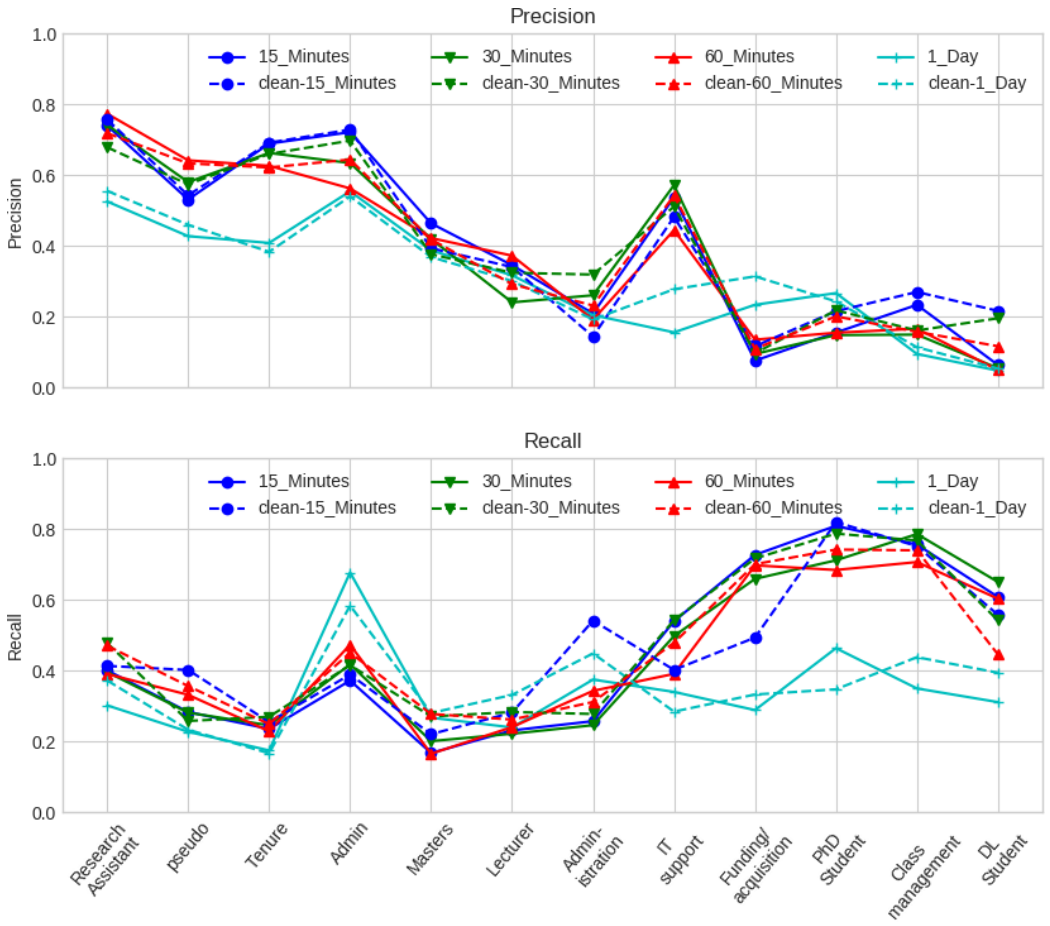


Figure 5.9. Port Distribution Set Precision/Recall Scores for SVM Classifier

Several observations can be made based on Figure 5.9. In general, as noted in Section 5.4.1

the relationship between precision values and role-group data set size was mixed, and the precision and recall values for the one-day slicing interval data tended to deviate significantly from the per-role average values of the other slicing intervals. The pseudo-group precision and recall values were, as found in Section 5.2 and Section 5.3, close to those of the Research Assistant and Tenure-role groups. As with the other SVM classification experiments, the average precision scores were higher than was observed using the nearest-centroid classifier. The cleaning of the data set did not appear to make a consistent difference in the recall or precision results.

5.5 Port Priority Vector Analysis

The data vectors used for this analysis consisted of port-priority vectors (Section 4.1.4). As described in Section 5.2.1, the Netflow data was processed for each combination of data cleaning (cleaned and not cleaned) and data slicing interval (15, 30, 60 minute and one day). Classification trials were repeated 10 times, and the averaged results used to create the plots in this section.

5.5.1 Nearest Centroid Classification

Figure 5.10 shows the nearest-centroid classifier precision and recall scores for each of the port-priority-vector data sets.

Several observations can be made based on Figure 5.10:

- In general, precision values decreased with role-group data set size. The average precision and recall values were noticeably lower than those observed with the other feature-vector data sets.
- Precision and recall measures based on the interval extremes (15 minute and one day) were more often the maximum or minimum values per role group. Measures based on the 30 or 60-minute slicing intervals tended towards the median value.
- Cleaning the data of automatic flows did not appear to create a consistent difference in the precision or recall measures relative to flow data that was not cleaned.
- The pseudo group had precision and recall values that were generally close to those of the Research Assistant and Tenure-role groups.

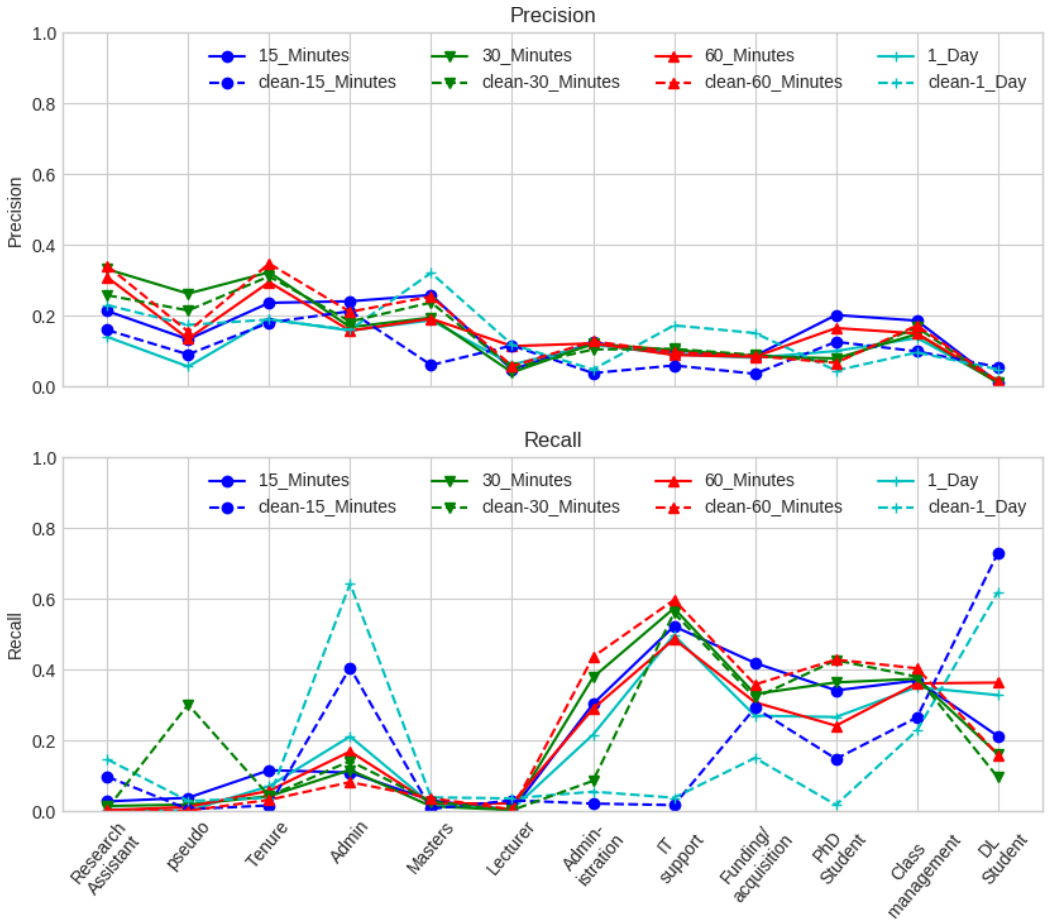


Figure 5.10. PPV Set Precision/Recall Scores for Nearest Centroid Classifier

5.5.2 Support Vector Machine Classification

Figure 5.11 shows the SVM classifier precision and recall scores for the port-priority-vector data sets. While average recall and precision values (with the exception of the pseudo group) were higher than found using the nearest-centroid classifier Section 5.5.1, the same observations apply to the SVM results.

5.6 User Class Consolidation

Because the results of our classification sets were visibly impacted by the unbalanced-data sets used to train the classifiers, we tested our classifiers on user group data sets approximately equal in size. To do this, we collapsed the groups listed in Table 3.3 into the larger categories of Faculty, Student and Staff. To equalize the group data set sizes, for

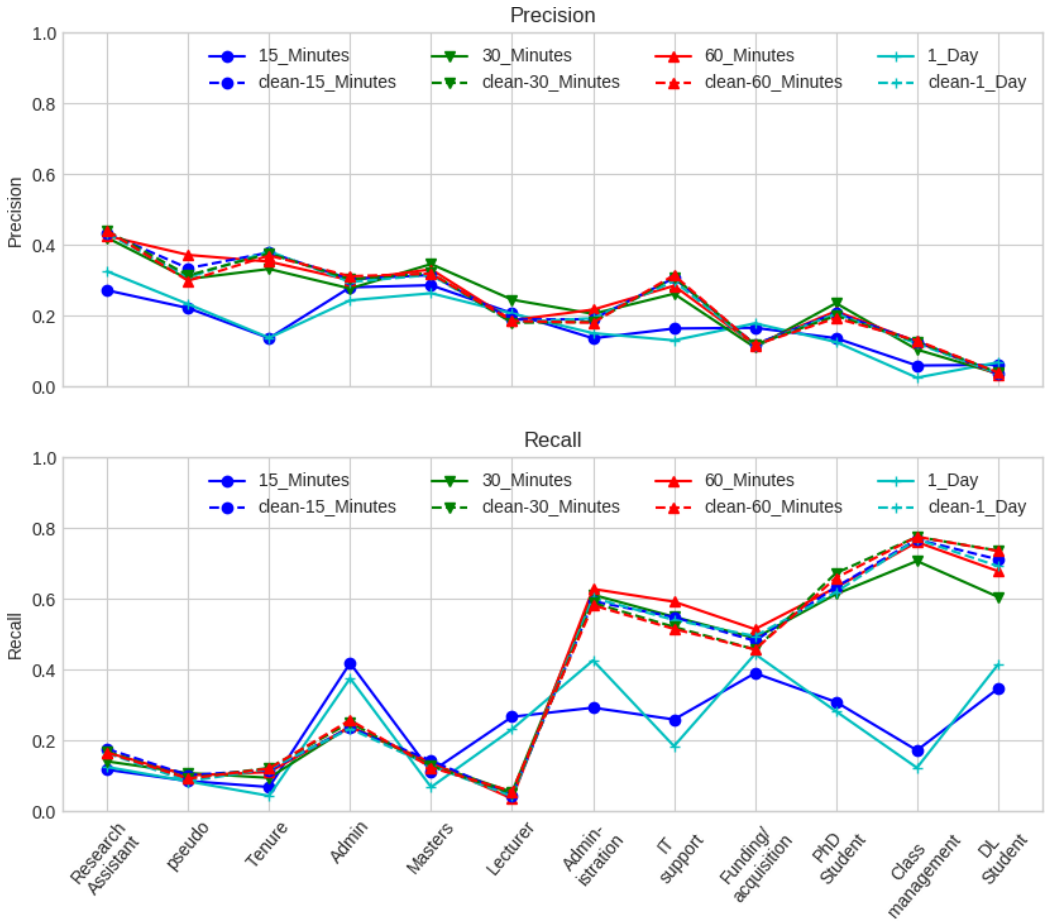


Figure 5.11. PPV Set Precision/Recall Scores for SVM Classifier

each feature-vector type we removed randomly selected user data sets from the larger two groups until the numbers of feature vectors for each group was within +/- 2%. To create the test (pseudo) group, users were randomly selected from the three groups and the associated data extracted. Because the intent was to create four balanced role-group-data sets, the user data sets extracted from each role group equaled 25% ± 1%.

Figure 5.12 shows the precision and recall scores for each role group based on training and testing the Nearest Centroid classifier on the baseline feature-vector-data sets. As can be seen in the figure, the classifier performed equally well on the Faculty and pseudo-group data sets, indicating that the precision and recall scores were primarily related to the type of feature vectors being processed and the size of the data sets.

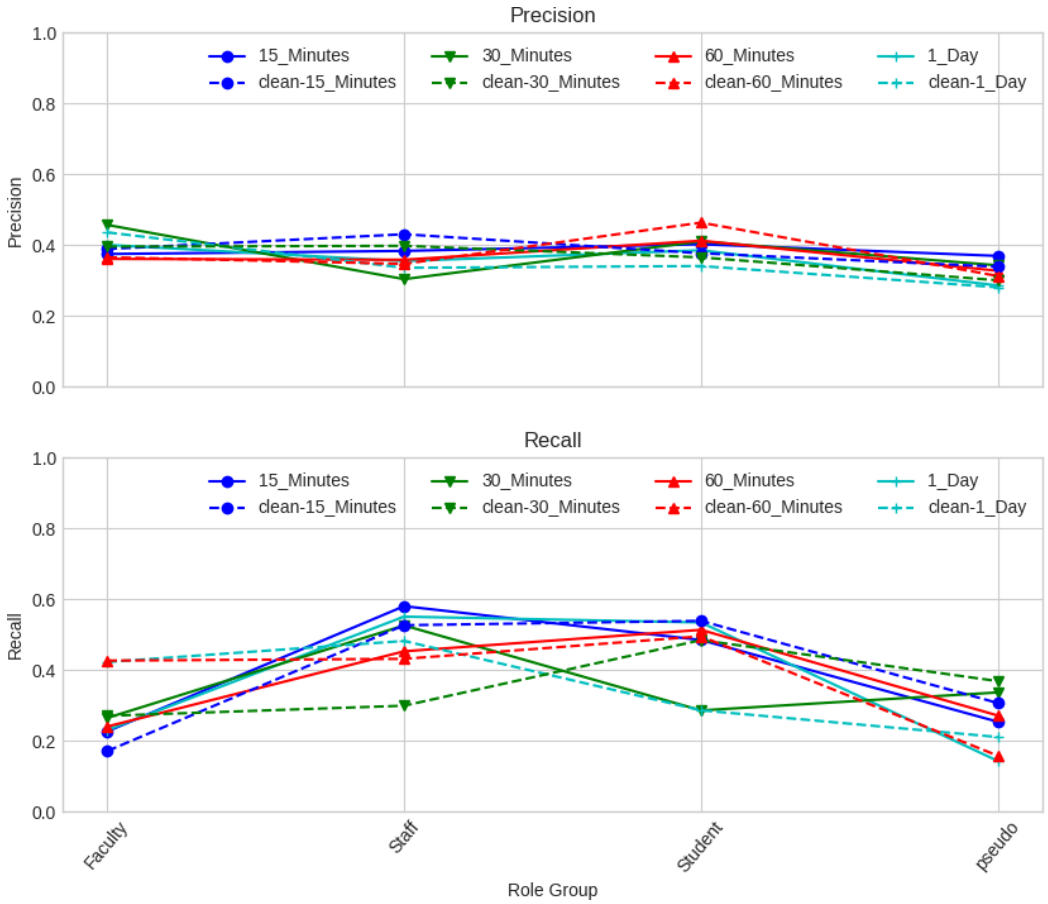


Figure 5.12. Baseline Set Precision/Recall Scores for Nearest Centroid Classifier - Consolidated Groups

A plot of the classifier performances on the consolidated-role groups for each of the feature-vector sets is presented in Section 5.7, based on the cleaned, 30-minute sliced Netflow data associated with the users in the role groups.

5.7 Feature Set Classification Comparisons

In sections 5.2 through 5.5, we tested the ability of our two classifiers to differentiate between data sets representing the network activities of 12 role-based groups, using four different feature vector types. The four feature vector types included feature sets based on aggregate statistical measures of Netflow record samples (see Table 4.4), statistical measures on the 15 port-protocol combinations listed in Table 4.3, byte-value distributions (see Section 5.4) for flows over the selected 15 port-protocol combinations, and port-

priority vectors (Section 3.2.2), which relate the ordered listing of observed port-protocol combinations in a flow data set to a global (all collected flow data) ordered listing of port-protocol combinations.

Figure 5.13 shows the precision and recall results of the classifiers for the feature vectors derived from cleaned data sliced on 30-minute intervals. Data derived from the 30-minute sliced data sets was used because the classifier results for data sliced at this interval showed fewer wide swings in precision and recall scores across the role-groups.

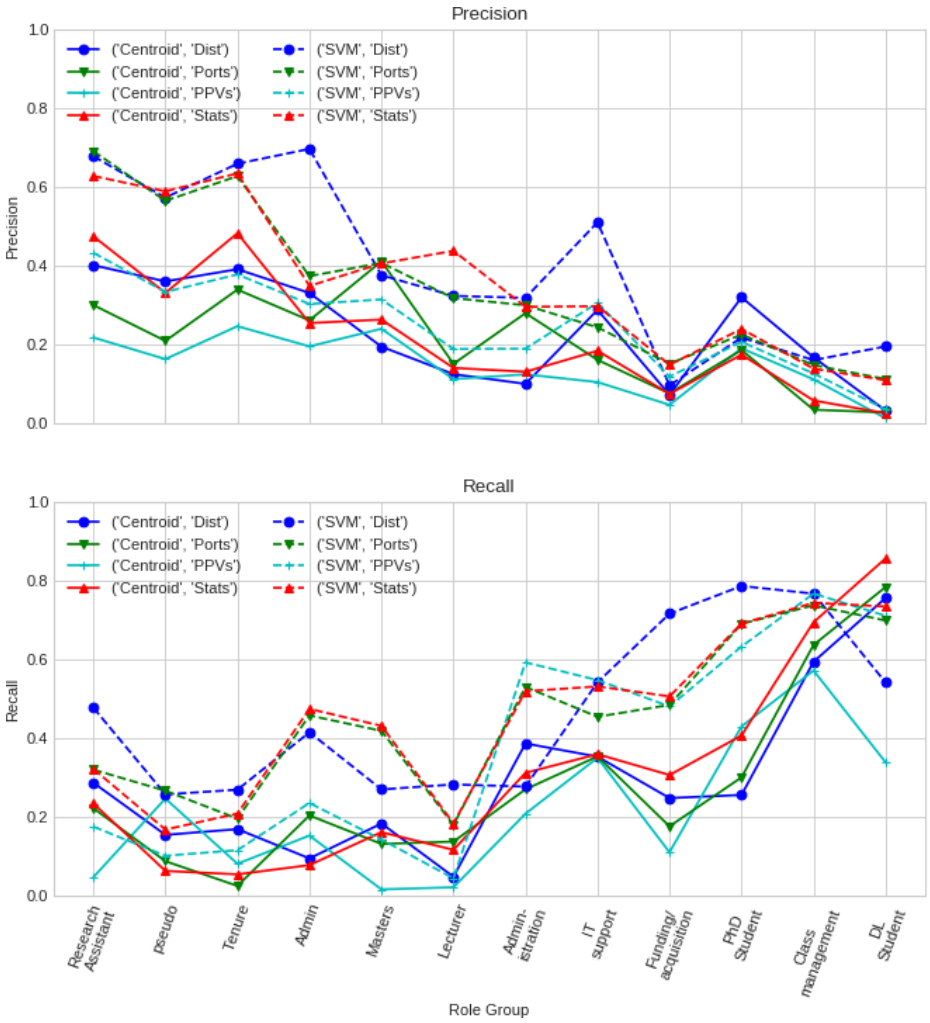


Figure 5.13. Comparison of Approaches

From this figure we can see that in general, for both the Nearest Centroid and SVM classifiers the smaller role-group data sets (DL Student, Class Management, Funding/Acquisition)

tended to show lower precision and higher recall values, while the larger role-based group data sets (Admin, Lecturer, Master's Student, Research Assistant, Tenure) tended towards higher precision and lower recall values. In both cases, these relationships between data-set class size and precision and recall values reflect the effects of training classifiers on unbalanced classes. Figure 5.13 also shows that in general the precision values of the pseudo-group were slightly lower the scores for the Research Assistant and Tenure-role groups, which were the closest in size in terms of total feature vectors per role group. Recall scores were approximately equal to those of the Tenure-role group.

Each of the classifiers was able to identify some fraction of the test data correctly, but it is difficult from the precision and recall graphs to determine which feature vector set provided the best classification results. Table 5.5 provides the mean F-scores obtained by the classifiers for each feature vector type. The values shown in Table 5.5 are the average F-score across each of the role groups tested (including the pseudo group). For our experiments, the Support Vector Machine with a radial-basis-function kernel performed on average better than the simpler Nearest-Centroid classifier. Both classifiers performed best with the feature vectors based on port-protocol byte-value distributions (Dist), and on cleaned Netflow records sliced over 30 or 60-minute intervals. Both classifiers also performed most poorly with the port-priority vector (PPV) data sets.

Table 5.5. Mean F-Scores vs. Cleaning and Slice Intervals

		Centroid				SVM			
Slice Interval	Cleaned	Stats	Ports	Dist	PPVs	Stats	Ports	Dist	PPVs
15_Min	No	0.168	0.141	0.182	0.122	0.290	0.306	0.324	0.175
15_Min	Yes	0.141	0.130	0.188	0.083	0.314	0.316	0.364	0.162
30_Min	No	0.169	0.147	0.195	0.096	0.282	0.293	0.316	0.206
30_Min	Yes	0.156	0.165	0.199	0.110	0.321	0.322	0.360	0.214
60_Min	No	0.159	0.162	0.185	0.103	0.281	0.293	0.319	0.215
60_Min	Yes	0.163	0.143	0.205	0.095	0.303	0.313	0.353	0.209
1_Day	No	0.145	0.140	0.147	0.094	0.247	0.237	0.285	0.213
1_Day	Yes	0.182	0.152	0.163	0.098	0.259	0.261	0.301	0.212

Table 5.6 shows the average F-score performance of each classifier on each of the feature vector types versus the role groups used for our analysis. Based on these scores, the classifiers consistently performed better data associated with the Admin-role group.

Table 5.6. Mean F-Scores vs. Role-Group

Classifier	Feature vector	Research Asst	Pseudo	Tenure	Admin	Masters	Lecturer	Administration	IT support	Funding/acq	PhD Student	Class mgmt	DL Student
Centroid	Stats	0.235	0.169	0.066	0.257	0.135	0.099	0.280	0.192	0.085	0.242	0.100	0.062
Centroid	Ports	0.189	0.107	0.051	0.241	0.174	0.109	0.241	0.224	0.095	0.222	0.057	0.057
Centroid	Dist	0.292	0.143	0.170	0.246	0.167	0.131	0.130	0.236	0.122	0.272	0.221	0.066
Centroid	PPVs	0.051	0.053	0.081	0.180	0.043	0.021	0.131	0.133	0.128	0.139	0.202	0.040
SVM	Stats	0.399	0.254	0.244	0.410	0.371	0.230	0.342	0.325	0.233	0.323	0.195	0.119
SVM	Ports	0.399	0.279	0.250	0.418	0.366	0.253	0.353	0.324	0.235	0.314	0.204	0.118
SVM	Dist	0.505	0.383	0.328	0.529	0.289	0.284	0.256	0.431	0.206	0.297	0.265	0.158
SVM	PPVs	0.220	0.141	0.149	0.279	0.174	0.112	0.271	0.330	0.201	0.280	0.173	0.079

Role-Group Consolidation: : Figure 5.14 shows the precision and recall scores achieved in classifying the different feature-vector-data sets for the equally sized consolidated-role group (Faculty, Student, Staff) data sets.

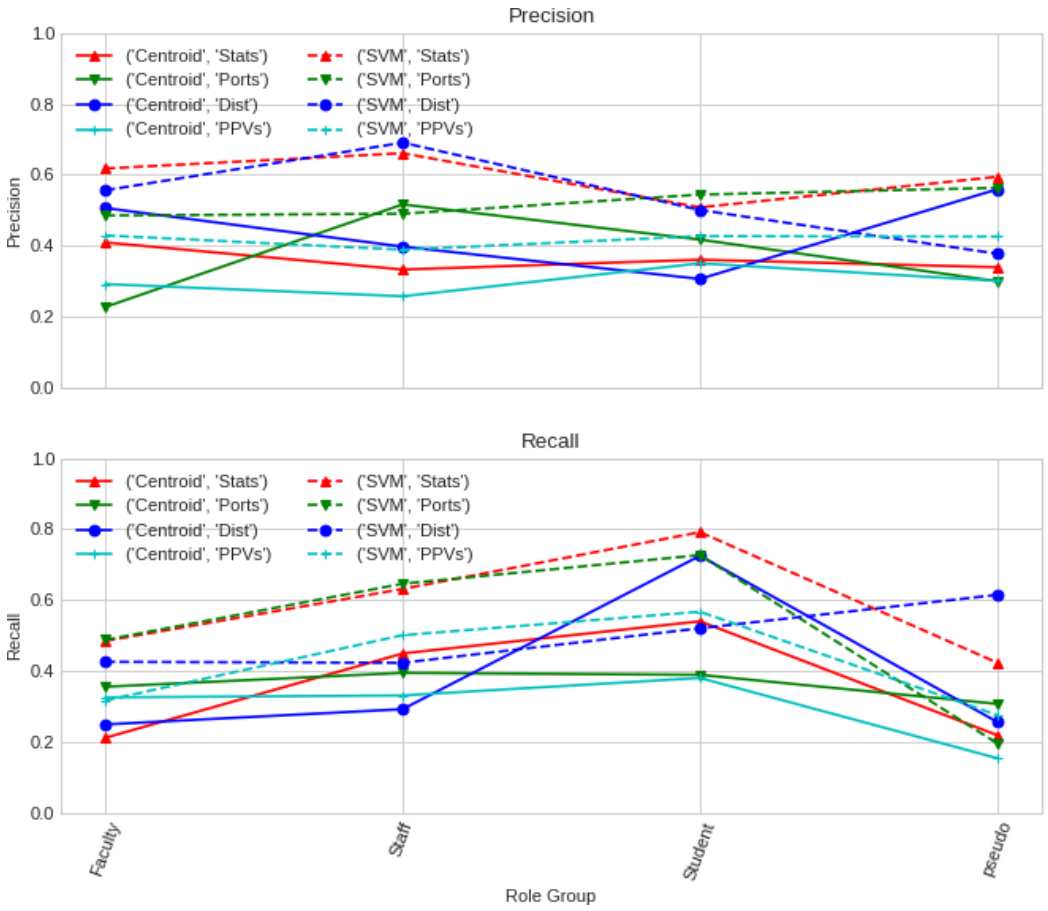


Figure 5.14. Comparison of Results - Consolidated Groups

The ranges of the precision and recall scores for the pseudo and Faculty-role groups were effectively the same, indicating that the scores were not based on how the role-group-data sets were formed, but instead were more directly related to the feature-vector type and the relative sizes of the data sets.

5.8 Clustering Analysis

To examine whether some fraction of flow traffic patterns generated by users are recognizably unique to their role group, we clustered data from each of the four feature-vector data sets using the K-means++ algorithm provided by the scikit-learn python library [63]. For each feature vector type, we clustered the feature vectors derived from cleaned Netflow record data sliced at 30-minute intervals. The number of clusters was set at 50, enough to discern whether some relatively pure (dominated by one role group) are formed. Figure 5.15 shows the results of clustering the baseline-feature-data set (discussed in Section 5.2.1).

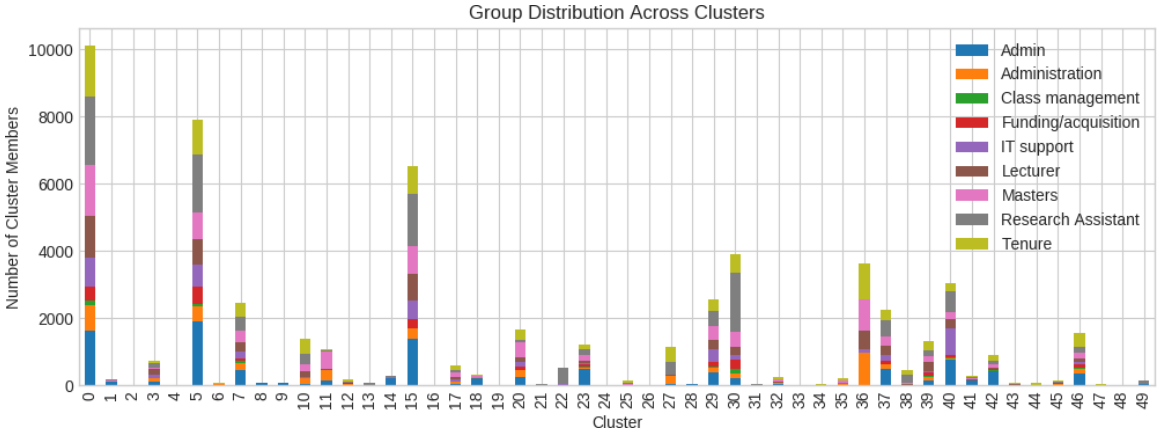


Figure 5.15. Group Membership of K-Means Clusters for Statistically Derived Feature Vectors

The larger clusters were quite mixed in terms of the role groups represented, with the largest role-group data sets (Masters Student, Research Assistant, Admin, Tenure, Lecturer) dominating the cluster memberships. In some of the smaller clusters the memberships were less mixed; clusters #8, #9, #14, and #28 consisted primarily of data vectors from the Admin role group and cluster #14 held data-vectors mostly associated with the Research Assistant role group.

Figure 5.16 shows the results of clustering the Port-Behavior feature-vector-data set. The

larger clusters are again highly mixed in terms of role-group membership, with membership proportions roughly correlating with role-group data set sizes. Of the smaller clusters, clusters #8, #10 and #23 primarily contain data vectors derived from the Admin role group while cluster #3 consists mostly of Admin-group-feature vectors.

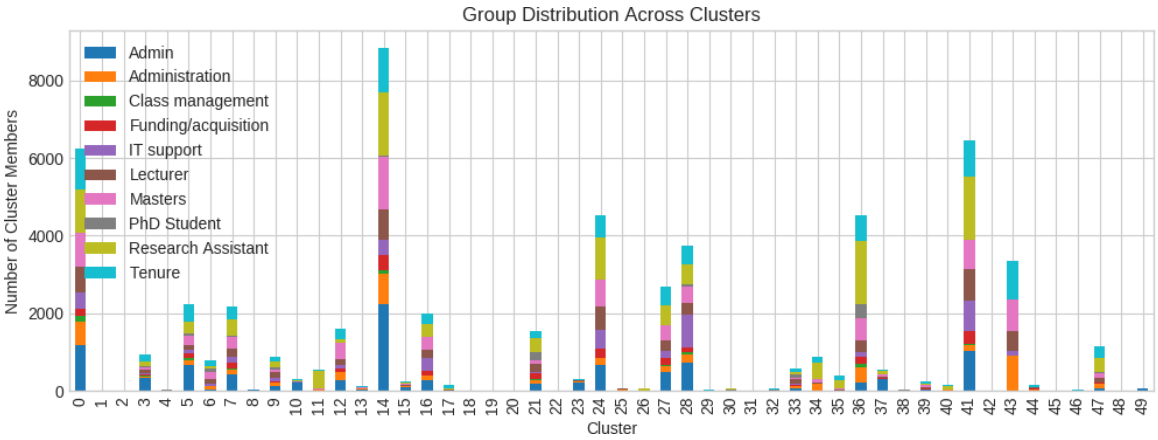


Figure 5.16. Group Membership of K-Means Clusters for Port-Behavior-Feature Vectors

The results of clustering the port distribution based feature vectors are shown in Figure 5.17. This data set continues the trend of creating larger clusters containing data-vectors derived from multiple role groups. The smaller cluster #16 contained a mixture of mostly Tenure and Research Assistant feature vectors.

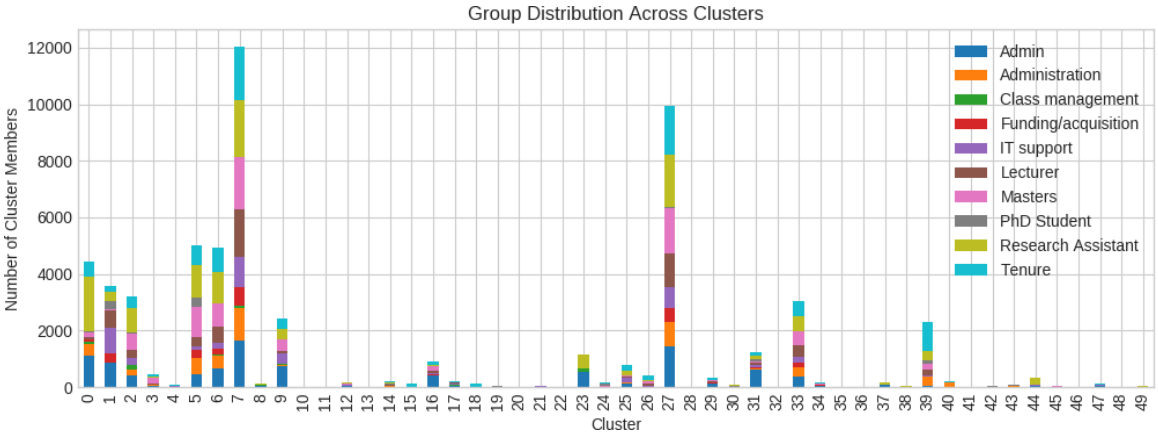


Figure 5.17. Group Membership of K-Means Clusters for Port Flow Distribution Derived Feature Vectors

For the PPV data (Figure 5.18), cluster memberships were more uniformly mixed for both

large and smaller clusters. No cluster was dominated by feature vectors from one or two role groups. Based on this, it is reasonable that a classifier would have a more difficult time differentiating role groups based on PPV data. This is born out by the performance of the classifiers on the PPV data. For the Nearest Centroid classifier, the correct role-group vectors were selected more often only three times for the cleaned, 30-minute-interval based data and four times for the non-cleaned data. Using the SVM classifier, the correct role-group vectors were selected more often (six of the 11 true role groups), for both the cleaned and non-cleaned data. These results represent significantly poorer classification performance than was achieved with the baseline-feature vectors evaluated in Section 5.2.2.

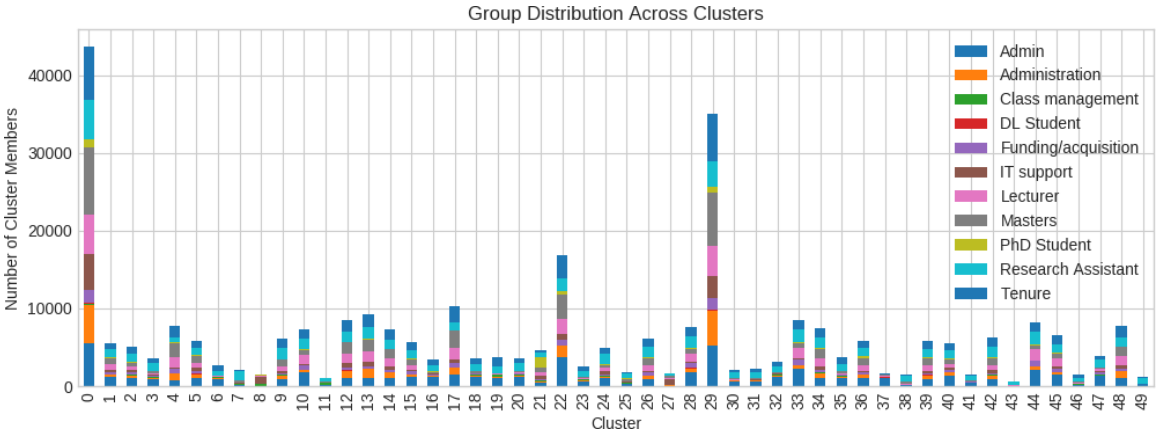


Figure 5.18. Group Membership of K-Means Clusters for PPV Feature Vectors

The fact that the clusters generated did not align well to specific role-groups is not a definitive measure of the lack of pattern similarities within role groups, however. These results show how the feature vectors we defined cluster together, using the K-means++ algorithm. Other combinations of features, shorter feature vectors less impacted by the curse of dimensionality, or other clustering algorithms could have shown different results.

5.9 Feature Vector Distance Experiments

Based on the results in Sections 5.2 through 5.5, there does not appear to be a strong relationship between user roles and the network traffic patterns they generate. Comparing the data sets of users directly however, to themselves (self similarity), to the users in their group (intra-group distance) and to users in other groups (inter-group distance), can provide

a different story. Distances between user data sets were determined by consolidating the feature vectors for each user and week of network activity into centroid vectors. Each centroid vector consisted of the mean feature values for a user during one week of activity (Section 4.4.3). Figure 5.19 shows the distance distributions observed when comparing the mean-feature-vector distances between user-data sets.

Self-similarity distributions were obtained by measuring the mean euclidean distances between the centroid vector set associated with each group user and itself. Distances between the same centroid vectors within a set (where the distance is zero) were excluded from determining the mean distance values. Intra-group distributions were created by computing the mean-pairwise distances between the data sets of users within the same user group. The remaining distributions shown for each role group are based on the mean-Euclidean distances between the data sets of users in the subject-role group (e.g. the Admin group for the first row in Figure 5.19) and the data sets of users in other role groups. The feature vectors used for this graphic were based on cleaned Netflow data sliced every 30 minutes.

Several observations are apparent. First, the self-similarity distributions had noticeably lower interquartile ranges than those of the intra-group and other-group distributions. This indicates that flow patterns generated by users tends to be consistent over time. In addition, for most of the role groups the mean feature-vector distances between users within the same role group were not significantly different from the distances to user-data sets in other role groups. The intra-group distributions associated with the PhD students and the Funding/Acquisition role group appear to be exceptions, probably due to these groups having the smallest data sets.

The distribution patterns observed in Figure 5.19 are repeated in Figure 5.20, which shows distance distributions obtained using port-behavior-feature vectors (Section 4.1.2). Likewise, Figure 5.21 shows self-similarity distributions are also lower for the byte-distribution-feature vectors (Section 5.4). The intra-group distances obtained using port priority vectors (Section 4.1.4) data were slightly lower than comparisons with role-groups (Figure 5.22). This supports the observation that port priority vectors were not as useful for machine-learning-related algorithms as compared to other feature vector types.

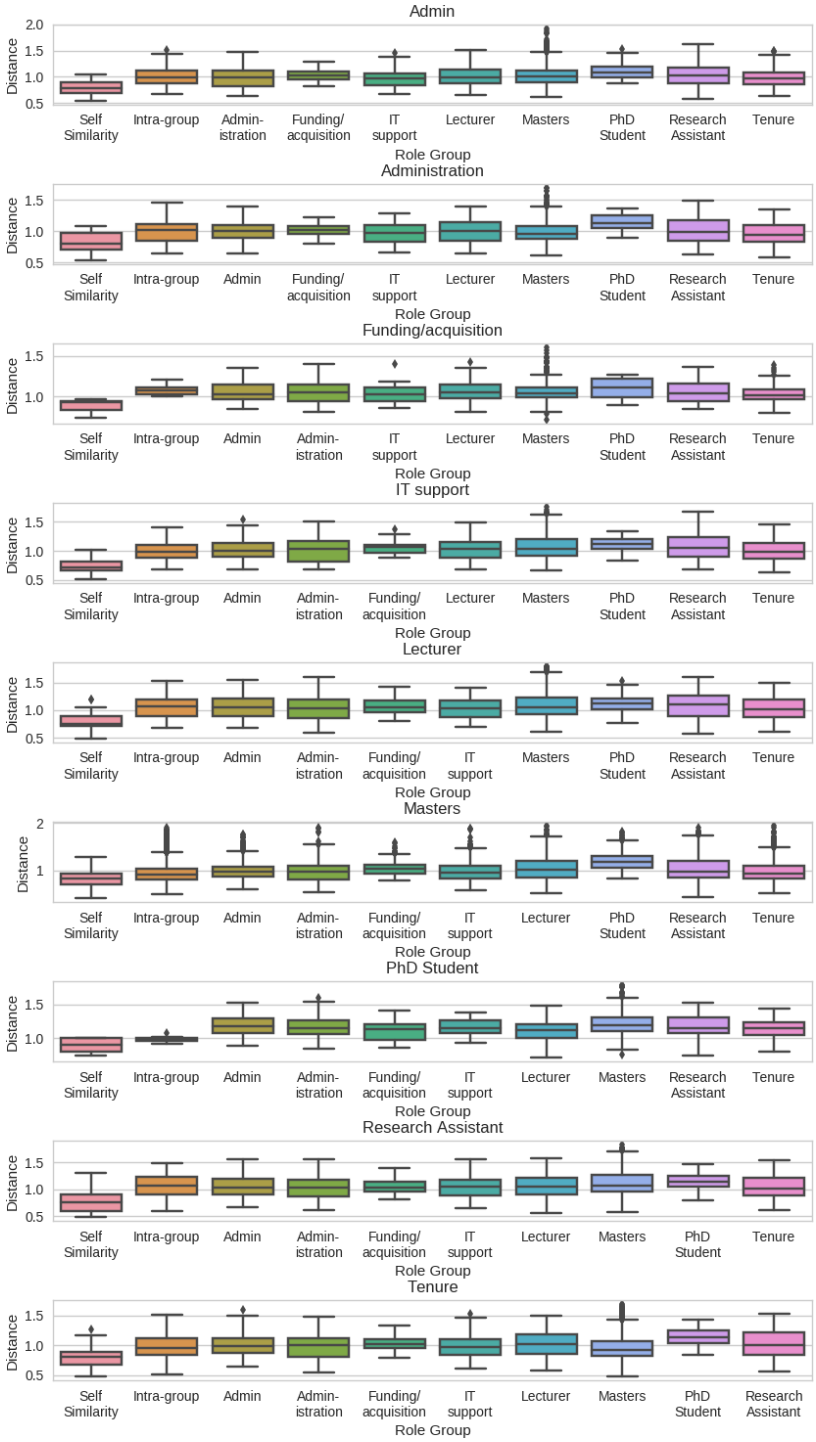


Figure 5.19. Self-Similar, Intra- and Inter-Group Distances for Aggregate Statistical Features

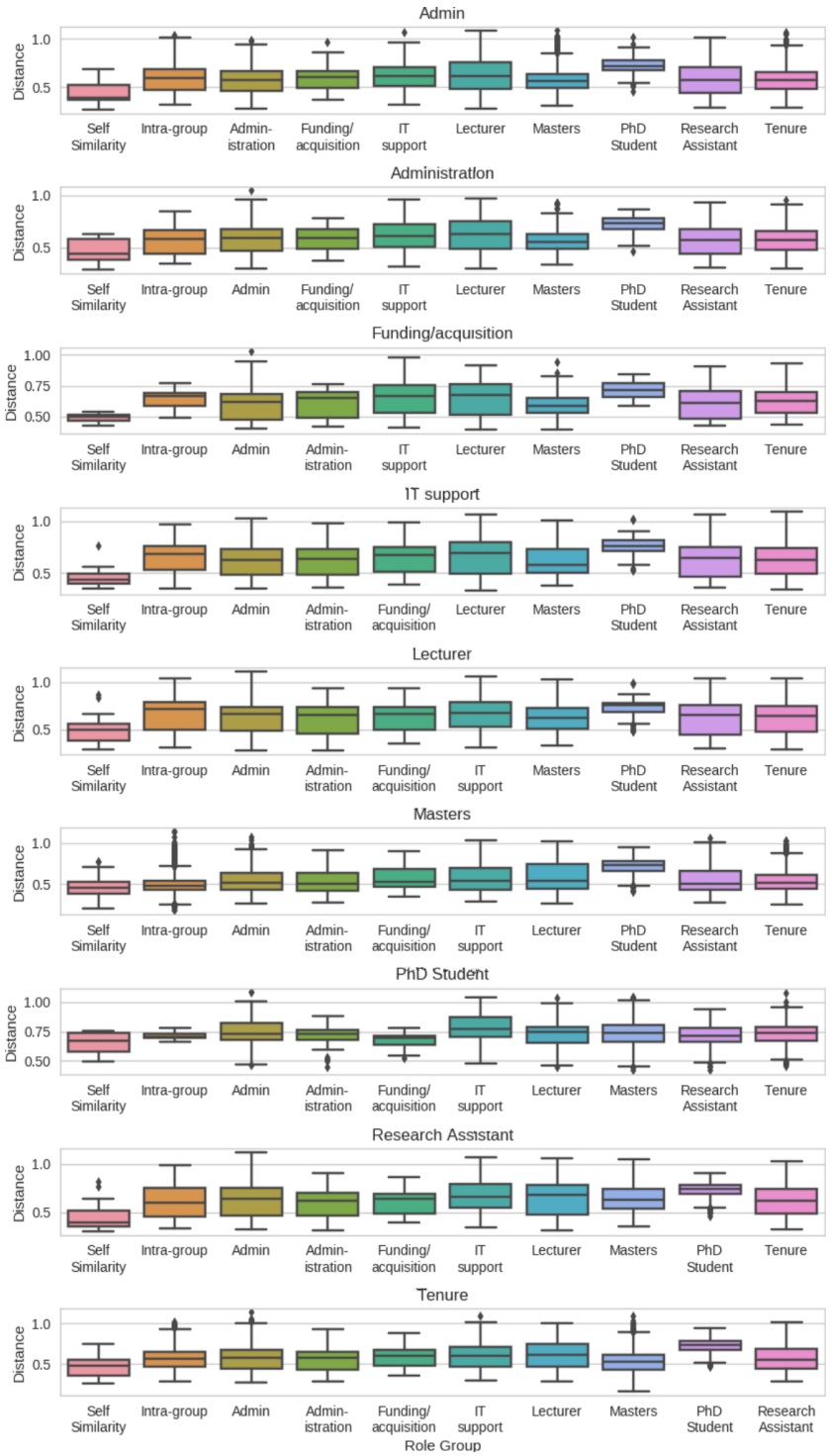


Figure 5.20. Self-Similar, Intra- and Inter-Group Distances for Statistical Port Features

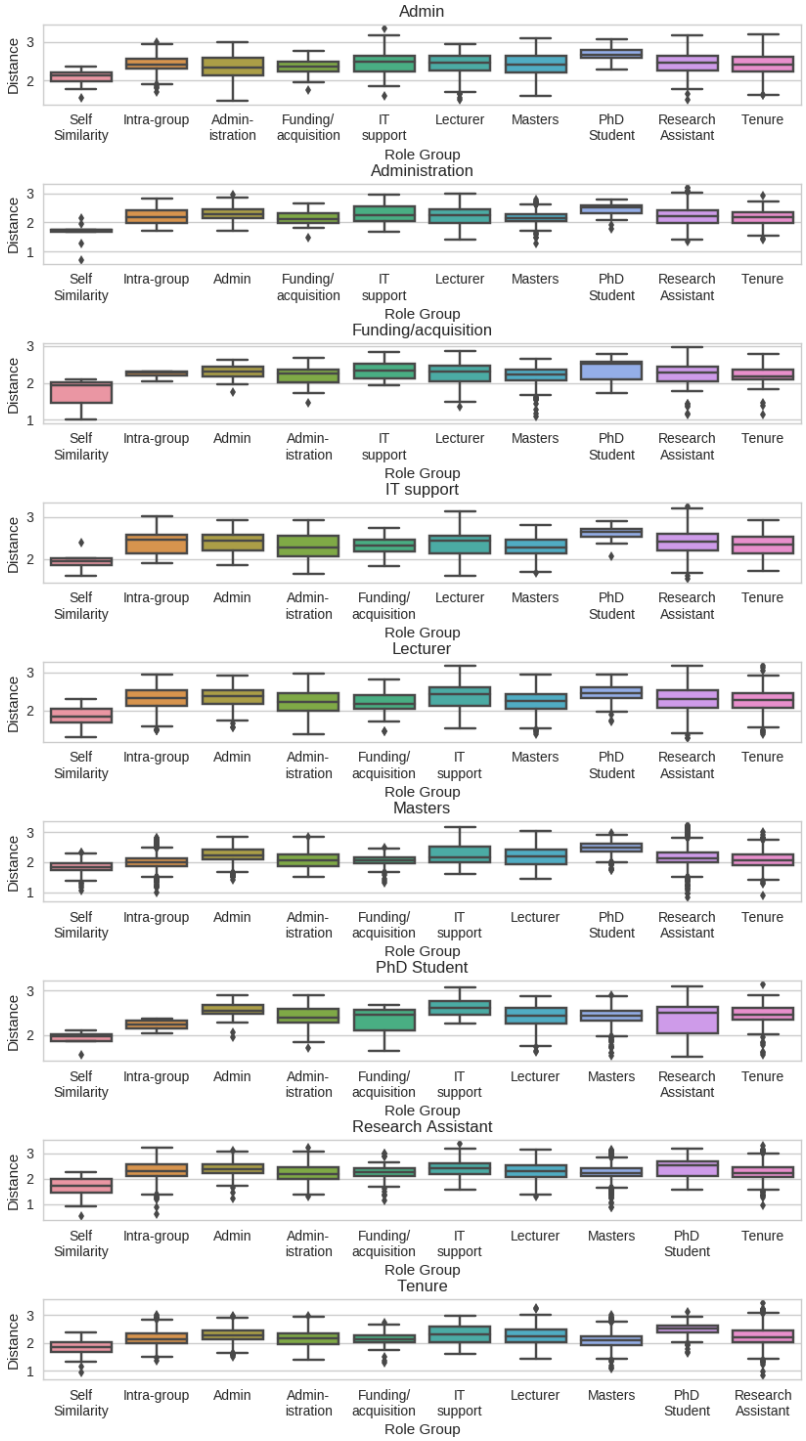


Figure 5.21. Self-Similar, Intra- and Inter-Group Distances for Port Byte Distribution Features

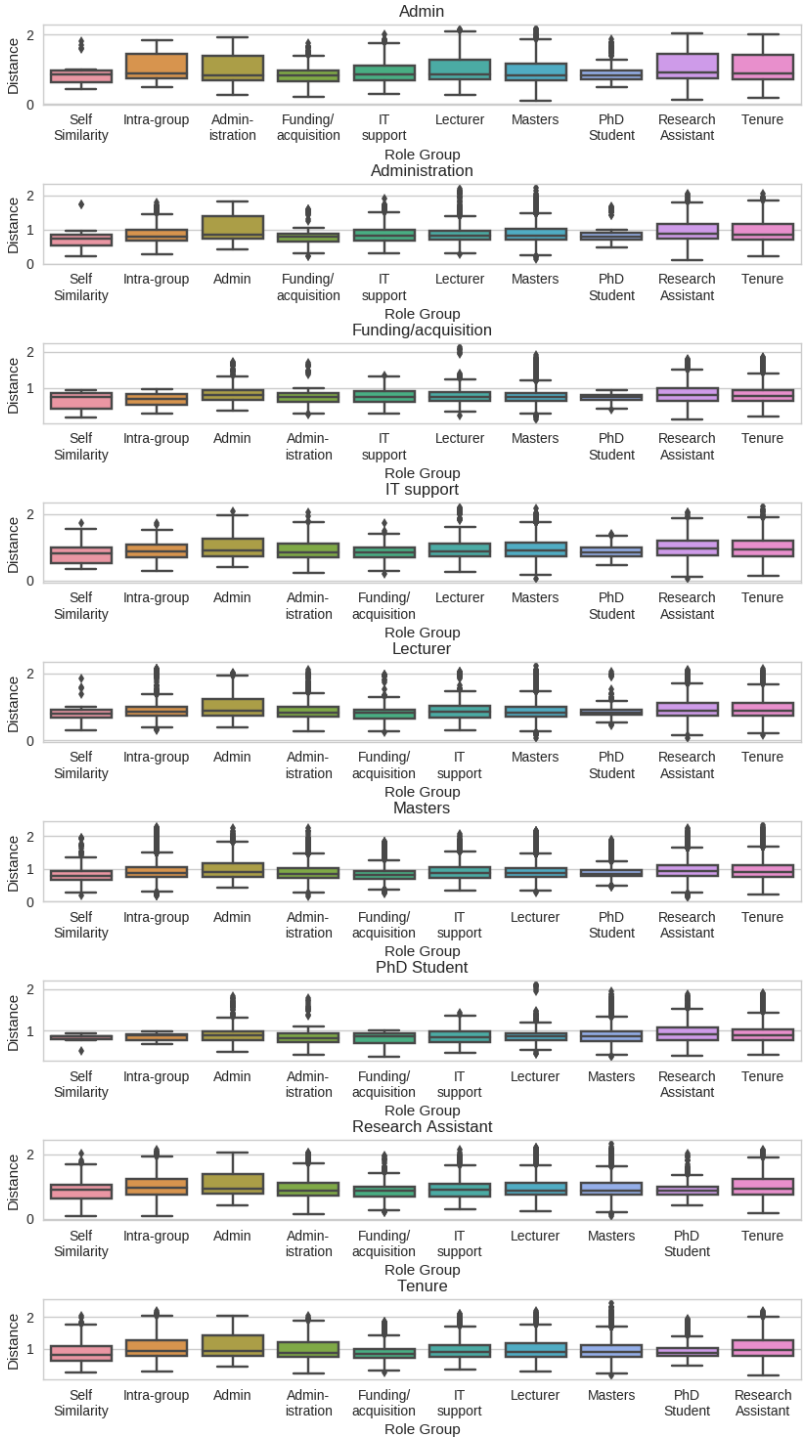


Figure 5.22. Self-Similar, Intra- and Inter-Group Distances for PPV Features

Figure 5.23 shows self-similar, intra- and inter-group distance distributions for equally sized, consolidated role-group data sets. The feature-vector type used for the distance comparisons in Figure 5.23 was the baseline (Section 4.1.1) set of features, derived from cleaned, 30-minute-sliced Netflow data. As observed in the other distance comparisons, the self-similar interquartile ranges were noticeably lower than the interquartile ranges for the intra- and inter-group data set distances.

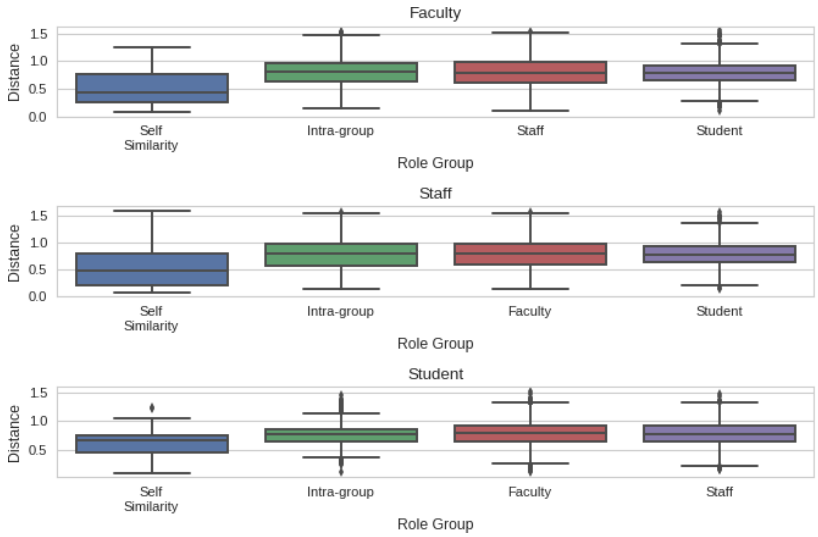


Figure 5.23. Self-Similar, Intra- and Inter-Group Distances for Aggregate Statistical Features

5.10 Grouping Users by Similarity

Based on the analysis thus far, we can conclude that the roles that we attributed to our users had little apparent effect on their network behaviors, as measured by the features we defined. The classifiers performed no better with data derived from our defined role groups than they did for the pseudo groups. The predicted similarities in behaviors by users sharing similar roles were not apparent in testing, based on any of the feature-vector-data sets. Clustering of the feature-vector-data sets created a few small role-group dominated clusters, but the majority of the clusters were highly mixed in terms of role-group representation. Finally, when the average distances between user-data sets were measured, in general distance comparisons of user-data sets within the same role group were no closer comparisons between user-data sets from different role groups. The only data sets that appeared to

exhibit consistent similarities were user-data sets compared to themselves.

To identify user groups that do contain users with similar network behaviors, we adapted a methodology described by Frias-Martinez [3]. We clustered the users based on the similarities between their feature-vector-data sets (described in Section 4.5). For a given set of feature vectors, the vectors are grouped based on the week the represented data was collected. For each user and week, centroid vectors (vectors of mean values for each feature in a feature-vector set) are calculated. The centroid vectors are clustered using K-means++, and user groups defined by which user centroids grouped in each cluster. A pseudo group is generated by extracting users and associated data vectors from the defined groups to test whether classifiers performed more poorly with a mixed-group set.

5.10.1 Classifier Testing of Behavior-Based-User Groups

The feature-vectors for each week were used to train and test a Nearest-Centroid classifier. To enable comparisons of the data clusters across each week of data evaluated, the center coordinates for the k clusters generated from the first week's data were used as the initial cluster centers for clustering data for the other weeks. While some drift in cluster centers was expected during the following weeks, we expected that this process would enable us to treat each cluster as representing similar behaviors over the data collection period. Table 5.7 shows the confusion matrix for the classifier tested on the baseline-feature set (Table 4.4), derived from cleaned, 1-day sliced Netflow data.

	0	1	2	3	4	5	6	7	8	9	10	pseudo	Recall
0	**15**	1	0	1	1	0	0	0	1	3	2	0	0.581
1	2	**10**	1	0	0	2	0	1	1	4	2	0	0.441
2	3	0	**31**	1	0	14	1	4	2	2	1	1	0.522
3	0	0	0	12	0	0	0	3	0	0	0	0	0.747
4	3	1	1	1	**32**	1	1	1	5	2	4	2	0.615
5	1	0	14	1	0	**73**	0	6	3	1	0	0	0.731
6	1	1	0	0	1	0	**5**	0	1	2	1	0	0.424
7	1	2	11	**21**	1	21	0	**17**	4	2	1	1	0.211
8	7	3	6	2	5	6	0	2	**26**	8	4	**3**	0.360
9	3	4	1	0	3	0	0	0	2	**28**	5	1	0.584
10	7	7	1	2	7	2	2	1	5	14	**29**	2	0.368
pseudo	3	2	6	4	4	9	2	3	5	5	3	1	0.026
Precision	0.319	0.335	0.435	0.274	0.590	0.567	0.455	0.266	0.474	0.393	0.556	0.098	

Table 5.7. Confusion Matrix For Clustered Groups

The number of clusters for this experiment was set to 11, to enable comparisons with the analyses using the 11 plus one (pseudo) role groups performed in Sections 5.2 through 5.5. The values in the table are averages, based on classification results for each of the five weeks of collected traffic. As can be seen in Table 5.7, the classifier performed significantly better in differentiating between the relabeled-user groups. Unlike the confusion matrices obtained by grouping users based on roles (Section 5.2), for 10 of the 11 new user groups the Nearest-Centroid classifier associated more feature vectors with the correct group (i.e. cluster) than it mislabeled from the other groups. The classifier performed most poorly with the pseudo group, which shows that data from the user groups identified by our clustering process were more identifiable than data associated with randomly selected users.

This pattern of higher scores was not followed for each of the slicing periods tested, however. For the 15 and 30-minute sliced, cleaned data sets, only four of the user groups had more of the correct user-group-data set selected, while for the 60-minute sliced, cleaned data sets this happened for eight of the 11 user groups. Figure 5.24 shows the precision, recall and F-scores for the baseline-feature set (Table 4.4), comparing the scores for the different slicing intervals.

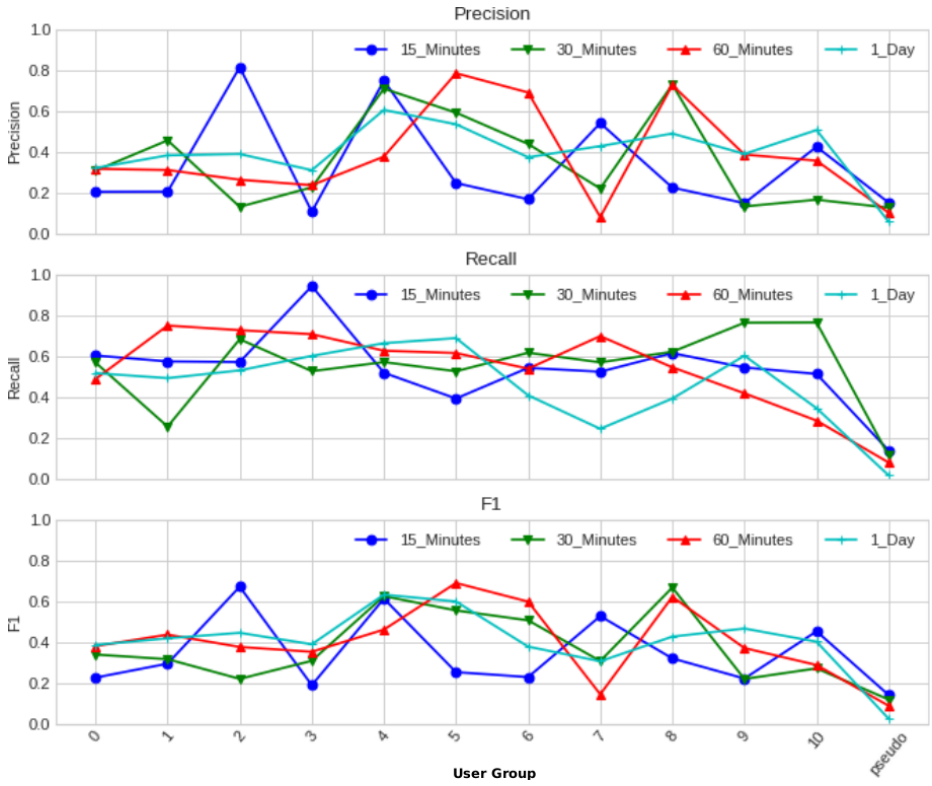


Figure 5.24. Clustered Group Scores For Intervals

While the precision, recall and F-scores for user data sliced over 1-day intervals remained fairly stable across the clusters, the values associated with the 15, 30 and 60-minute intervals fluctuated significantly. We attributed this trend to the fact that the clusters used to define user groups were created based on feature values averaged over a week. Slicing flow data over shorter periods means that any feature vectors generated reflect short-term, transient behaviors, and thus feature values would show more variability as compared to feature vectors based on longer slicing periods. Greater feature value variability would mean more vectors would fall at greater distances from the weekly mean values. Because the classifier performed best using feature vectors created from flow data sliced over one-day intervals, the rest of this section will present results based on feature vectors created from cleaned flow data sliced over that interval.

Figure 5.25 shows the precision, recall and F-scores for tests of the Nearest-Centroid classifier on each of the four feature-vector types described in Sections 5.2 through 5.5. For these tests, the port-behavior (Section 5.3) and port-distribution (Section 5.4) based

feature vectors showed the greatest variability in scores, while the baseline (Table 4.4) and PPV (Section 4.1.4) based feature vectors were the most stable in value across the different clusters. The pseudo group again showed the lowest average precision, recall and F-scores as compared to the clustered-user groups, showing that the classifier did not find common behaviors within the pseudo group.

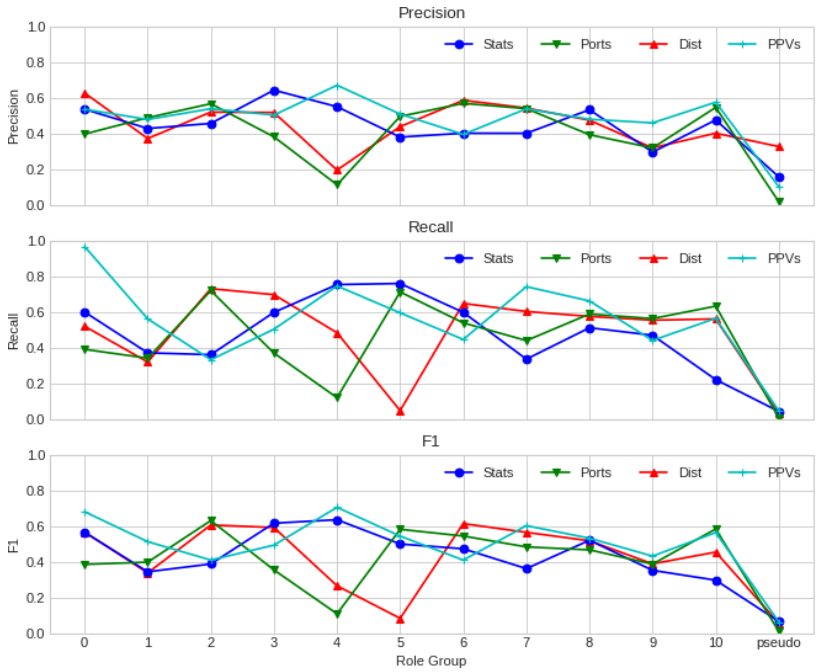


Figure 5.25. Precision, Recall and F-scores of Clustered Groups For Each Feature-Vector Type

Finally, to test the effect of creating fewer user groups, consolidation was performed by reducing the number of clusters. Figure 5.26 shows the precision, recall and F-Score measures obtained for classification based on five user groups, for vectors derived from 1-day sliced, cleaned Netflow data. As observed in Figures 5.24 and 5.25, the pseudo group received the lowest average scores. For this smaller set of user groups, only the distribution-based feature-vector set displayed significant variation in the recall values for the different user groups.

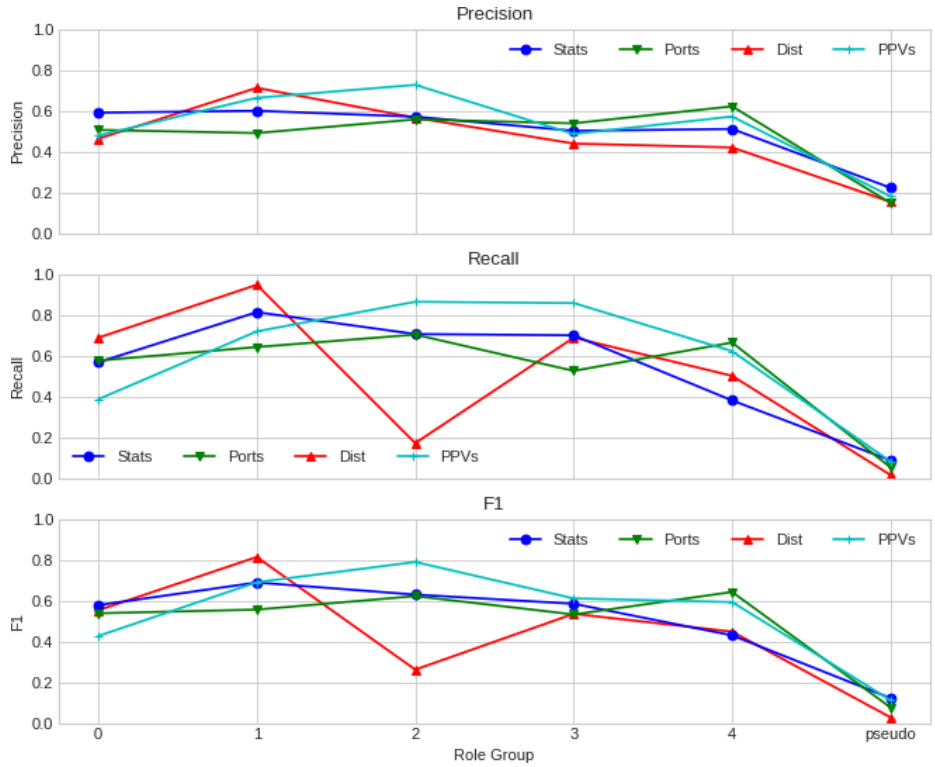


Figure 5.26. Precision, Recall and F-scores For Five Clustered User Groups

The average F-score for the five-user group classification analysis was higher than was achieved using 11 user groups. This indicates that the number of clusters used can be tuned to optimize feature-value-detection thresholds for each user group, to enable the detecting anomalous feature values. In the access-control system described by Frias-Martinez [3], she performed cross-validation tests on a subset of user profiles, varying the number of clusters used to group users in order to find a k value that resulted in the most true-positive-anomaly detections and the fewest false-positive results.

For a direct comparison of the differences between user groups created via role labels or by behavioral similarities, we compared precision and recall scores directly for three user groups (without adding control groups). Figures 5.27 and 5.28 show the precision and recall scores achieved with the nearest-centroid classifier for data derived from user groups created using behavior-based and role-based grouping, respectively. The precision and recall scores for classifying data from the behavior-based user groups were distinctly higher than was obtained with role-based grouping. Figure 5.27, depicting the behavior-based

group scores, only show scores for feature vectors derived from one-day slices of user Netflow data because scores for shorter intervals showed greater variation (as shown in Figure 5.24). The baseline-feature set (Table 4.4) was used for this comparison, derived from cleaned Netflow data.

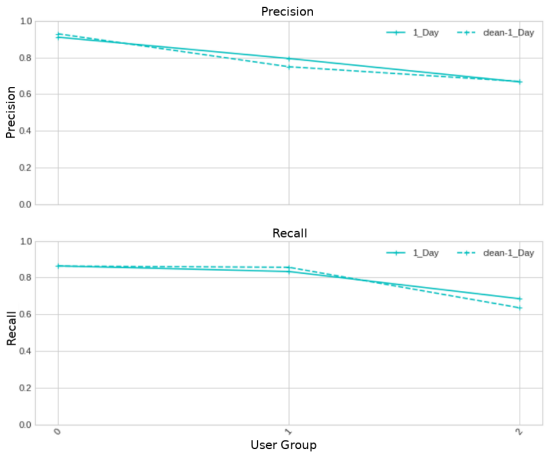


Figure 5.27. Clustered User Groups Scores

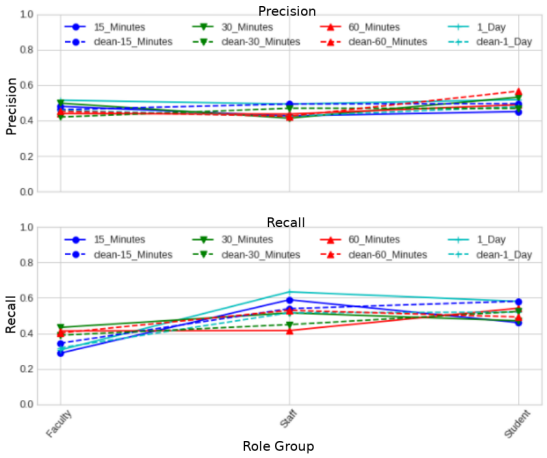


Figure 5.28. Role-Group Scores

5.10.2 Clustering Behavior-Based-User Group Data

Because user groups were defined based on behavioral similarities, we expected clustering of the relabeled feature-vector-data sets would result in clusters dominated by one or two user groups. Using K-means++ we clustered feature vectors derived from cleaned, one-day sliced Netflow data; k was set to 50 for clustering the data.

Figure 5.29 shows the cluster memberships obtained for feature-vector data derived from cleaned, one-day sliced Netflow data, representing one week of collected network traffic. As was done with the classifier experiments, we defined 11 user groups based on behavioral similarities prior to clustering the data. The clusters consisted primarily of feature vectors from four user groups, indicating that the user groups generated were imbalanced in data-set size.

In Figure 5.29 we can see that a number of the smaller clusters generated consisted almost entirely of feature-vectors from one user group (designated by the number 2). These clusters appear to be proportionally larger than the single-role group dominated clusters shown in

Figure 5.15. This pattern of cluster sizes and memberships was observed for each week of collected data.

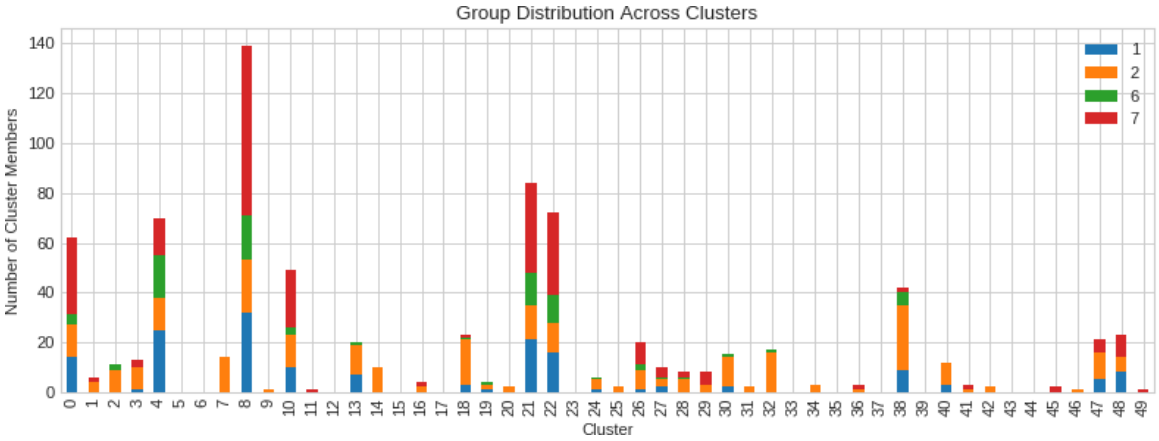


Figure 5.29. Cluster Membership for Behaviorally-Defined User Groups - Baseline-Feature Set

Figure 5.30 shows cluster memberships for one week of port-behavior-vector data. For this feature-vector set seven of the 11 created user groups were large enough to be visible in the graph, indicating imbalanced-group-set sizes but less so than observed with the baseline-feature set. With this data set we see the clusters containing primarily one user group are fewer and generally smaller than was observed in Figure 5.29. Compared to the single-group clusters generated using role-based labeling of the data however (Figure 5.16), there are more of them and they are proportionally larger.

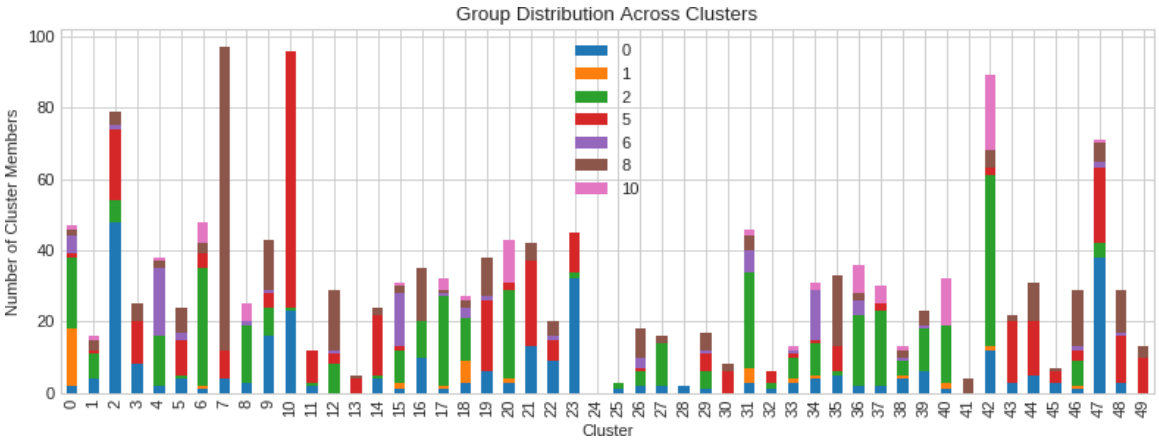


Figure 5.30. Cluster Membership for Behaviorally-Defined User Groups - Port-Behavior-Feature Set

Clustering results for the port-distribution feature-vector set is shown in Figure 5.31. Five of the 11 user groups contained enough user data to be displayed, again indicating imbalanced user-group sizes. The small, single-user-group-dominated clusters generated however are proportionally larger than the clusters generated using role-based-group labels (shown in Figure 5.17).

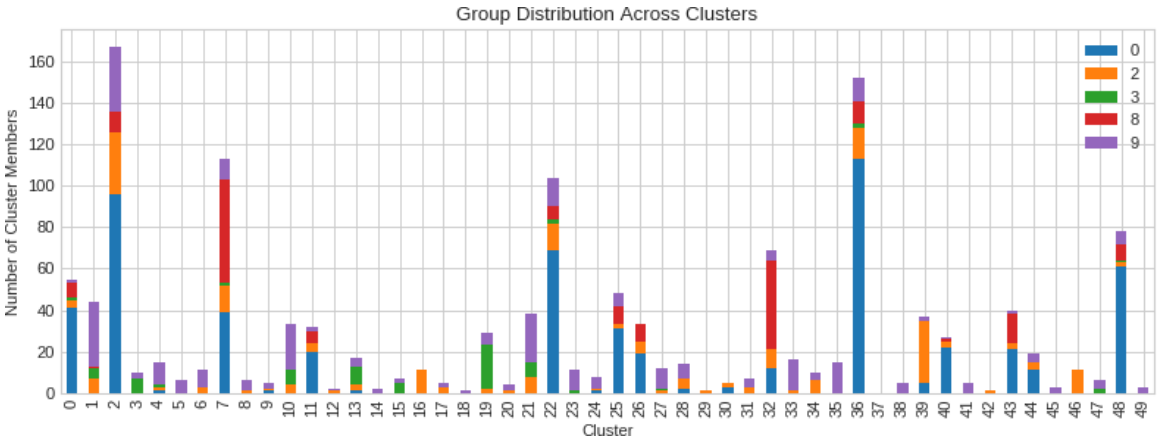


Figure 5.31. Cluster Membership for Behaviorally-Defined User Groups - Port-Distribution-Feature Set

Figure 5.32 shows the results of clustering the relabeled port-priority vector (PPV) data set. The user-group-data sets are again unbalanced, with only four user groups containing enough feature vectors to be visible in the graph. While the clusters are not as highly mixed in membership as was observed using the role-labeled PPV data (Figure 5.18), this may be due to the reduced number of effective user groups used.

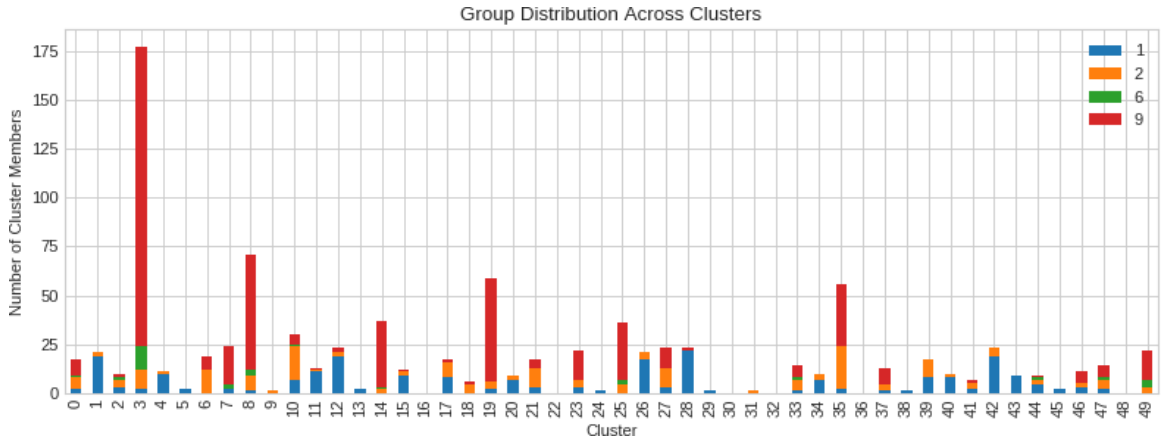


Figure 5.32. Cluster Membership for Behaviorally-Defined User Groups - PPV-Feature Set

5.10.3 Feature Vector Distance Comparisons

In comparing the mean distances between user-data sets, we defined user groups based on creating centroids of each user's data over the entire collection period and clustering them using K-means++. Averaging user-feature-vector values over the entire collection period was necessary, because the distance comparison method we used was based on comparing centroid vectors for each week of user activity (Section 5.9). If user groups were defined on a per-week basis, each user would have one centroid vector per week to compare with others and self-similar distance measurements would not be possible.

Figure 5.33 shows the self-similarity, intra- and inter-group distance distributions obtained by comparing data-set distances with the relabeled-user groups, using baseline-set (Section 5.2) feature vectors derived from cleaned, 1-day sliced flow data. As was observed in the distance comparisons using the role-labeled data we can see that the self-similar interquartile distance ranges are again lower than the inter- or intra-group ranges, but in most cases the intra-group interquartile distance ranges are a close second. This pattern is repeated in the distance comparisons for the port-behavior-data set (Figure 5.34) the port-distribution-data set (Figure 5.35), and the PPV-data set (Figure 5.36).

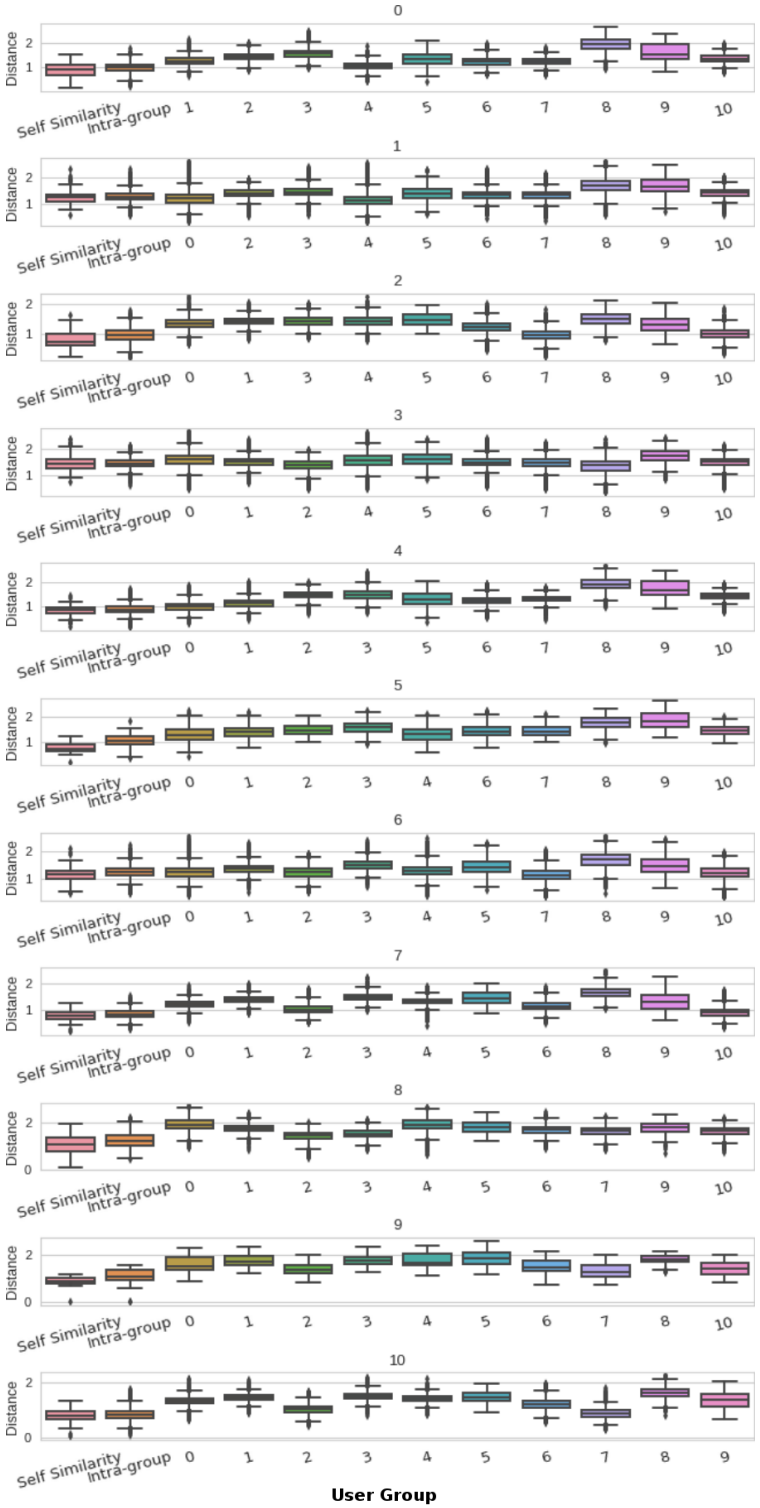


Figure 5.33. Self-Similarity, Intra- and Inter-Group Distances For Behavior Groups - Baseline Features

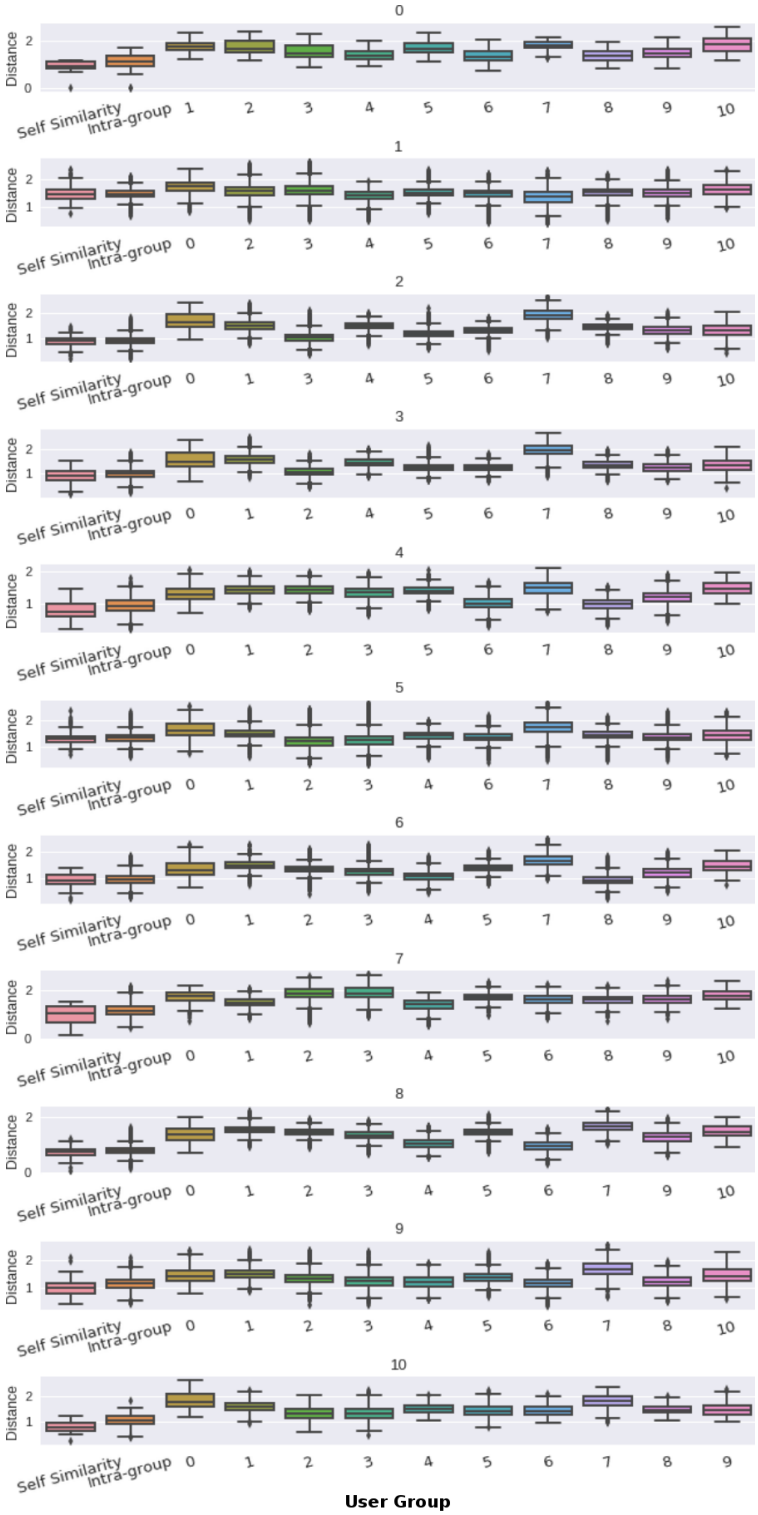


Figure 5.34. Self-Similarity, Intra- and Inter-Group Distances For Behavior Groups - Port-Behavior Features

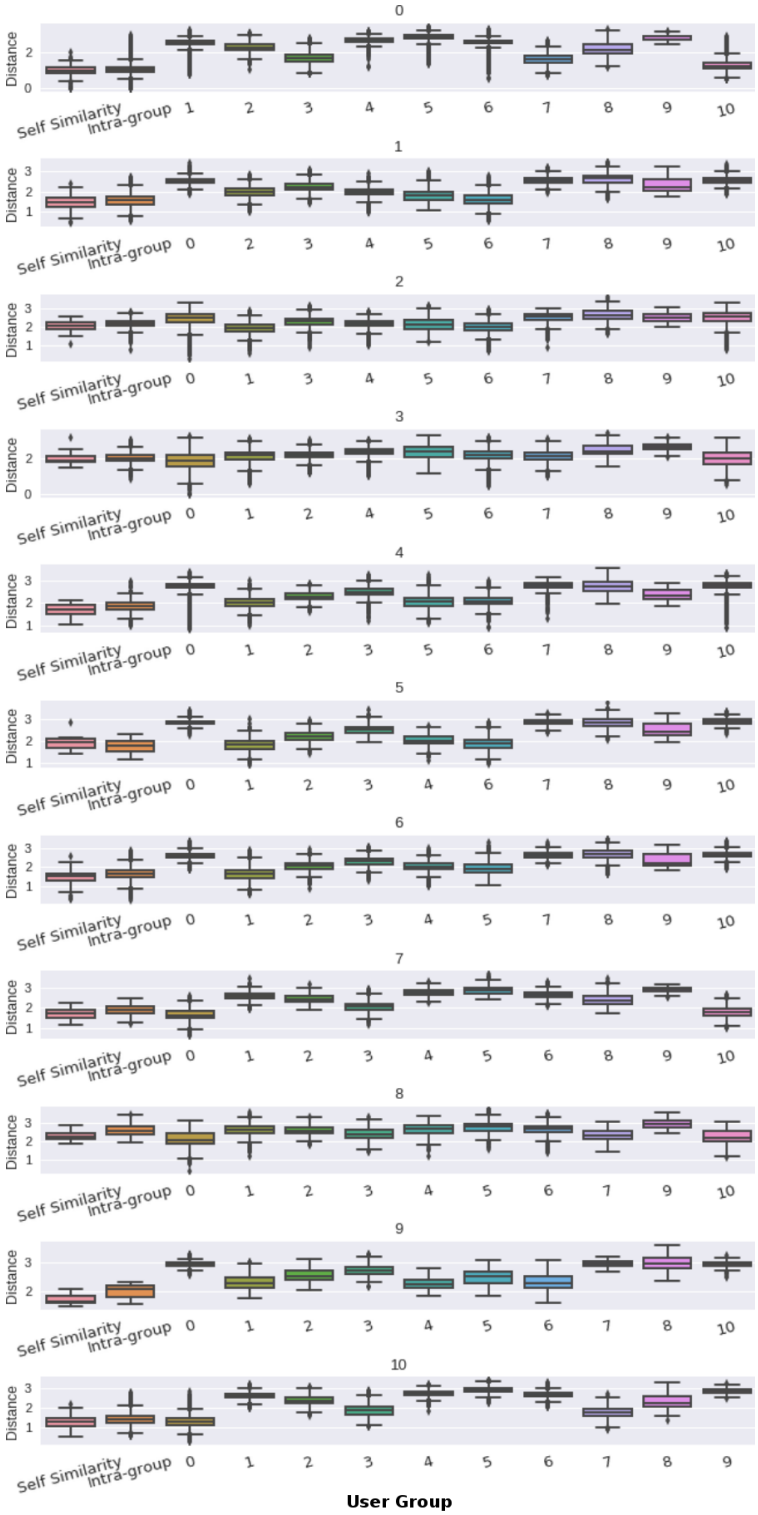


Figure 5.35. Self-Similarity, Intra- and Inter-Group Distances For Behavior Groups - Port-Distribution Features

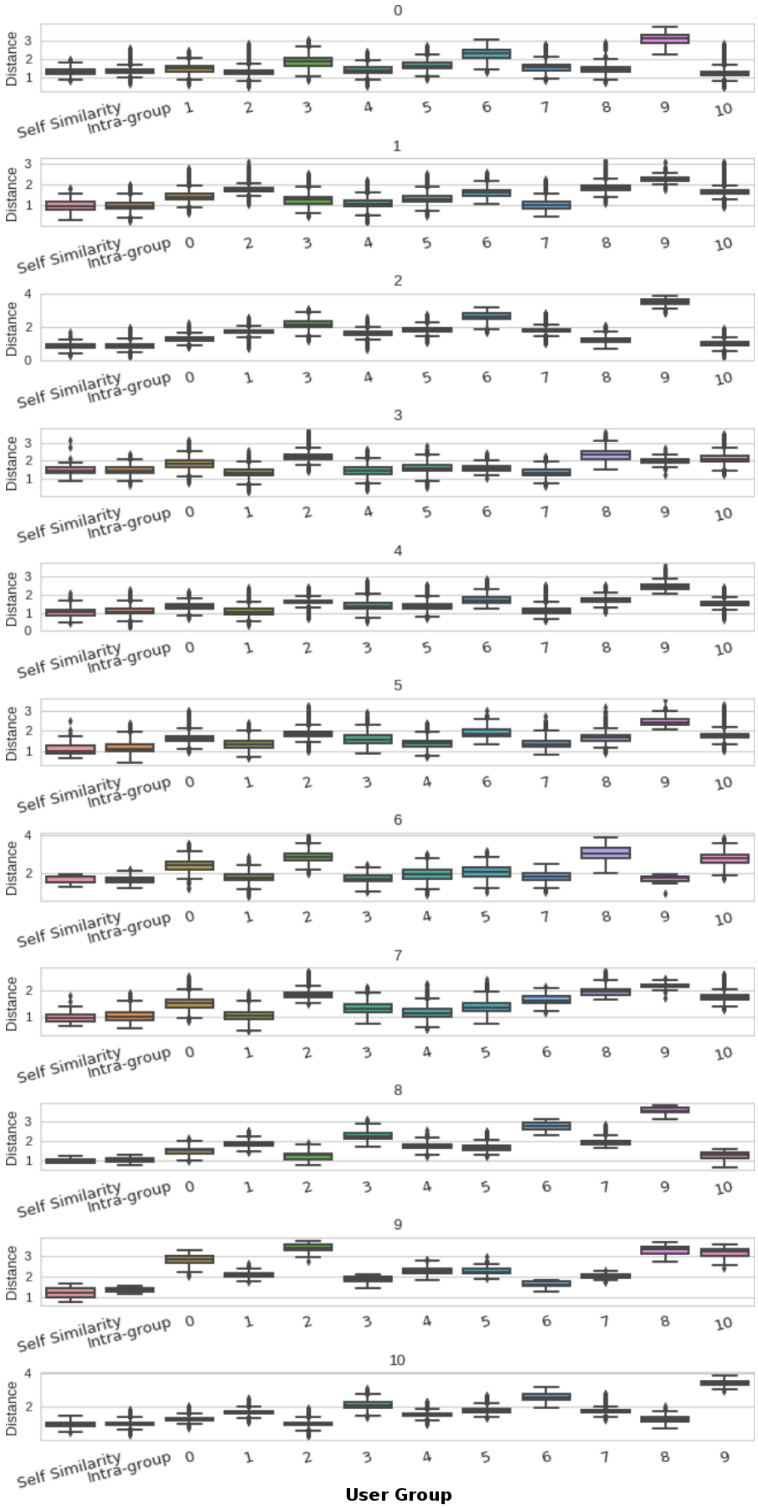


Figure 5.36. Self-Similarity, Intra- and Inter-Group Distances For Behavior Groups - Port Priority Vectors

The changes in intra- and inter-group distance ranges obtained by regrouping users based on behavioral similarities show that the user groups are now measurably different, something that was not apparent when the user data sets were grouped based on roles.

5.11 Conclusion

This chapter presented the results for the experiments discussed in Chapter 4. We compared the differences in the user-group characteristics obtained when grouping users based on their organizational roles to the characteristics of user groups generated by grouping users based on behavioral similarities. Comparisons were performed based on how well machine-learning classifiers differentiated between defined user-group-data sets and data sets compiled from randomly selected users. In addition, data sets were clustered using K-means++ to determine if any user group data clustered separately from the data of other role-groups.

To measure the effect a user's role has on the Netflow records they produce, we created four different sets of Netflow derived features, to characterize the network behaviors of users within each role-defined groups. As an experimental control, we created non-role related (pseudo) groups by randomly extracting users and their associated data sets from each role-group-data set. We trained and tested two different classifier algorithms to differentiate between the role-based and pseudo groups, to determine if the classifiers performed differently with the psuedo group that the did with the role-based groups.

Both classifiers were able to associate more feature vectors to the correct role group than to any of the other role groups. For the pseudo-role-group data, both classifiers consistently achieved precision and recall results similar to that achieved with similarly sized role group data sets. The radial basis function kernel SVM performed better than the Nearest Centroid classifier, and each classifier performed best on feature vectors representing traffic over selected port-protocol combinations using byte-value distributions.

In the course of testing this linkage between user roles and Netflow data using classifiers, we also evaluated the effects on classification results from varying how flow data using different port-protocol combinations is represented, varying the slice periods used to create the feature vectors, and the removal of flow data generated by automatic processes.

Clustering of feature-vector-data sets showed that in general the larger clusters contained feature vectors drawn from most of the role-group data sets, while some smaller clusters were dominated by data derived from one or two of the larger role-group data sets. This indicated that much of the flow patterns observed in the collected data was similar across the different role groups, with only small subsets of flow data characteristic of specific roles. The exception to this observation were the clusters generated based on Port Priority Vectors (Section 5.5), in which no one role-group dominated any clusters.

Comparing the mean euclidean distances between the feature-vector-data sets associated with each user showed that the interquartile range (IQR) of the distances between each user's data set and itself were consistently lower in comparison with user-data sets in any role group, including the user's role group. In addition, the intra-group IQR for each role group were mostly equivalent to the inter-group distance IQR.

The conclusions that can be drawn from these experiments are that roles have little impact on flow-data patterns captured as Netflow-derived-feature vectors. The flow patterns generated by the users in our study were mostly similar across role-groups, and the only source of behavioral consistency we observed was within the flow traffic generated by individual users.

These results were contrasted with results obtained by creating user groups based on measured behavioral similarities. Users were grouped by similarity by using K-means++ to cluster centroid vectors, each vector consisting of the mean feature values for a given feature-vector set and period of time. After clustering, user groups were designated by the cluster number a user's centroid vector was a member of. Using this approach to define user groups, the tests used to measure the effects of user-role groups on network traffic patterns were repeated.

The results obtained for the behavior-defined-user groups were measurably different from that of the role-defined-user groups. For the classifier tests, the Nearest-Centroid classifier consistently scored the pseudo group lowest in terms of precision, recall and F-scores. Clustering the relabeled data sets resulted in more clusters dominated by feature vectors derived from one of the user groups. Comparisons of the mean Euclidean distances between user feature-vector sets showed intra-group IQRs lower than the inter-group IQRs for most user groups, indicating that user data sets within each user group were indeed similar relative

to the other user-group data sets.

CHAPTER 6:
Conclusion

This dissertation presents an approach for testing the collective network behaviors of groups of users, and determining whether users in the groups demonstrate shared patterns of network behavior. It also compared the behavioral similarities of users in groups defined based on their organizational roles with the behavioral similarities of users groups defined based on sharing similar patterns of network behavior. This analysis was performed using flow metadata (i.e. Netflow) to capture the network activities of the users involved in this study. The process presented represents a wide look at how network flows as summarized by Netflow can be characterized through the selective use of groups of features, and successfully used to demonstrate measurable relationships between users.

6.1 Dissertation Summary

We presented our methodology for reducing the number of flows in the collected data set not relevant to an analysis of user behaviors on the network. Applying this methodology, our cleaning processes reduced the number of flows to store and analyze by 68.5%. After identifying the collected flow records that could be attributable to 1373 users on the network, the users were grouped in into 11 roles based on their positions at the Naval Postgraduate School. The records were then anonymized by replacing user names with ID numbers.

We developed several algorithms to recognize flows generated by systems automatically. We identified a defining trait of automatic flows, high repetition of flow related features, and created algorithms to identify and count the times such features appear within a set of flow records. Those flows associated with outlier-count values for the selected features were labeled as automatic. Identifying another automatic flow process, the periodic reloading of web pages, was performed in a similar manner. Flow-data sets where these automatic flows were removed were referred to in this dissertation as being cleaned.

The features we generated for comparing behaviors within different user groups focused primarily on network services associated with known ports and protocols. We created four different types of feature-vectors, one based on a broad set of Netflow-derived statistical

and information-theoretic measures, and three based on different approaches to describing flow activity over selected port-protocol combinations. The three port-protocol focused approaches characterized flows using statistical measures of activity, byte-value distributions, and Port Priority Vectors (Section 4.1.2). For each user we created multiple data sets of each feature-vector type, drawing from a user's Netflow data both with and without removing automatic flows as well as varying the data slicing rates used to create the feature vectors. The total number of combinations ([cleaned versus unfiltered flow data] x [data sliced over 15, 30, 60-minute and one-day intervals]), created eight data sets per user per feature-vector type.

We tested for the existence of role-group effects on user network traffic by applying two machine-learning classifiers (Nearest Centroid and Support Vector Machine) to the feature-vector data. The Nearest Centroid classifier provided a linear discriminator comparing each user's feature vectors against the mean feature values observed for each role-group class. The Support-Vector Machine (SVM) classifier operates by identifying boundaries between data points from different classes, where a boundary is positioned at the greatest distance between data points from a class and the rest of the data set. Our SVM was used with a non-linear kernel, enabling searches for an optimal boundary over higher dimensions, where more separation between data points from different classes may be found.

To serve as a control group for our tests, we randomly selected users from each role-group and relabeled their feature vectors as belonging to a pseudo-group class. The maximum fraction of feature vectors we extracted from any role-group was approximately 15%. We tested and trained the nearest centroid and support-vector machine (with a non-linear radial-basis-function kernel) classifiers on data sets for each feature vector type, with and without automatic flows removed, and for each slicing interval. For each trial, for each feature-vector type, slicing interval and cleaning state, the classifiers performed about as well identifying feature vectors associated with the pseudo group as they did with other role groups of similar size. These results indicated that grouping data sets based on user-role groups did not provide much apparent advantage in comparing user network behaviors.

The classification tests also demonstrated that the classifiers performed best in distinguishing between our role-group-data sets with the port-protocol byte distribution based feature vectors, and most poorly with the port priority vectors (PPVs). These results indicate that

using byte-value distributions to describe flow data provide more discriminating power for classifiers than the other feature vector formats we tested. The classifiers also performed more consistently on feature vectors derived from data sliced at 30 and 60 minute intervals, while feature vectors created from data sliced every 15 minutes or every 24 hours frequently resulted in precision or recall scores higher or lower than the median score values for a role group. Also demonstrated was the impact of cleaning the data of automatic flows; the impact was indeterminate. Cleaning did not consistently result in either higher or lower precision or recall scores.

We also performed K-means++ clustering on each feature-vector type to determine if data from any role group clustered separately from data from other role groups, which could indicate feature-vector patterns unique role-group behaviors. For each of the feature-vector types, role-group membership (number of feature vectors per role group) of the larger clusters were mixed, usually in proportion to the relative sizes of the role-group data sets. A few smaller clusters consisted primarily of feature vectors from one or two of the larger role groups. These results imply that in large part flow patterns created by members of each role-group are similar, with relatively small subsets that may be unique to certain roles. Of the feature-vector data types that were clustered, the PPV feature vectors resulted in highly mixed memberships for all the clusters generated. This indicates that classifiers would find it more difficult to discriminate between data from different role groups based on PPV features, which was consistent with what we observed.

In our final set of experiments on role-group labeled data, for each feature-vector type we grouped the data by user and the week of data collection. For each user and week of data, we created centroid vectors (vectors of mean feature values). With these centroids, we performed pairwise distance comparisons of data sets for each user, computing the mean euclidean distance between the centroid vectors in the user-data sets being compared. We found that on the average for each user, the closest match for their data sets were to their own data sets. In other words, on average a user's historical patterns of network activity were consistent across the five weeks of collection, and this consistency was reflected in the lower feature-vector distances within a user's data set. In addition, we found that on average each user's feature-vector-data set was no closer to the data sets of users in their own role-group than they were users from other role groups. This indicates that the only consistency we observed in our experiments for comparing user behaviors was from the

users themselves, and not the role-group box they might be put into.

The implications of these experiments are that a user's role within an organization does not have a measurable impact on the patterns of network traffic they generate, as documented in Netflow records. Similarity in behaviors appears to occur at the user level. We next created user groups by clustering user-centroid vectors, and grouping users based on the clusters their centroid vectors belonged to. We repeated the tests performed on the role-group-labeled data, although for these experiments we did not use the Support Vector Machine classifier.

The results obtained for the behavior-defined-user groups were measurably different from that of the role-defined-user groups. For the classifier tests, the Nearest-Centroid classifier consistently scored the pseudo group lowest in terms of precision, recall and F1-scores. Clustering the relabeled data sets resulted in more clusters dominated by feature vectors derived from one of the user groups. Comparisons of the mean Euclidean distances between user feature-vector sets showed intra-group IQRs lower than the inter-group IQRs for most user groups, indicating that user data sets within each user group were indeed similar relative to the other user-group data sets.

These results indicate that the data sets reflecting the behaviors of users grouped through clustering were similar, far more so than the groups defined by user roles.

6.2 Future Work

Establishing that users act more as individuals than as members of a role category provides a basis for refining improved methods of detecting anomalous behaviors among users. The consistency of the results we observed using multiple forms of feature vectors indicates that the relationships we explored between Netflow records, users and role groups will show up for different feature sets. Frias-Martinez [3] used features drawn from flow activity over a few ports to create a behavior based access control system, while we tested feature vectors based on a much larger set of port-protocol combinations. There is a trade-off space that can be explored using this approach, in terms of the number and combinations of features that can be most effectively used in profiling users, and in terms of the sizes of the profile clusters that would work best for setting alarm thresholds.

The performance of the classifiers using port priority vectors could be improved by exploring other possible prioritization schemes. Our ranking of the observed port-protocol flows was based on the total flows per port-protocol combination. Ranking could also be performed based on the total bytes passed, mean bytes per packet, mean packets or bytes per second, average number of packets per flow, or other possible measures. These alternative methods, tested individually or combined into longer feature vectors, may provide better resolution in terms of distinguishing different patterns of flow activity.

One possible big improvement to distinguishing between user patterns of behavior would be to improve the detection of automatically generated flows. Approaches to this would include identifying periods when the user is not present on the system. An idle system left overnight still generates network traffic. While the algorithms we developed would detect many of these automatic flows, recall for these algorithms was not 100%. Being an active campus with students often staying late to perform experiments, it was not feasible to excise Netflow data based on the time of day it was collected. A reliable method of detecting user presence at the computers would help significantly in reducing the automatic flows in the data.

6.3 Conclusion

The primary hypothesis for this dissertation was that the roles a user holds within an organization has an impact on the network flow patterns a user generates, and that this impact can be detected and measured. Our experimental results show this to not be a valid hypothesis, and that comparing network behaviors at the level of users (or of groups of users exhibiting similar behaviors) would be a more fruitful approach to detecting behavioral anomalies than using role groups as behavioral standards. Our research also provides some indications of the kinds of Netflow derived features in which the differences between user network behaviors is most visible. While much work has been done in the area of measuring user computer activities for the detection of anomalous behaviors and in the application of user roles for bounding normal computer usage, this is the first such study we are aware of that focuses on the relationship between user roles and their network activity as measured using Netflow.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] K. Downer and M. Bhattacharya, "BYOD security: A new business challenge," in *Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1128–1133.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[3] V. Frias-Martinez, "Behavior-based admission and access control for network security," Ph.D. dissertation, 2008, AAI3333480.

[4] G. F. Anderson, D. A. Selby, and M. Ramsey, "Insider attack and real-time data mining of user behavior," *IBM Journal of Research and Development*, vol. 51, pp. 465–475, May 2007.

[5] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[6] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: A survey," *Journal of Big Data*, vol. 2, no. 1, p. 3, 2015.

[7] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[8] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (IDPS)," *NIST Special Publication*, vol. 800, no. 2007, p. 94, 2007.

[9] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1, pp. 18–28, 2009.

[10] A. Lazarevic, V. Kumar, and J. Srivastava, "Intrusion Detection: A Survey," in *Managing Cyber Threats*. Springer, 2005, pp. 19–78.

[11] C. Gates and C. Taylor, "Challenging the anomaly detection paradigm: a provocative discussion," in *Proceedings of the 2006 Workshop on New Security Paradigms*. ACM, 2006, pp. 21–29.

[12] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *Security and Privacy (SP), 2010 IEEE Symposium on*, may 2010, pp. 305–316.

[13] A. K. Ghosh, A. Schwartzbard, and M. Schatz, "Learning program behavior profiles for intrusion detection." in *Workshop on Intrusion Detection and Network Monitoring*, 1999, vol. 51462, pp. 1–13.

[14] D. J. Hand and D. J. Weston, "Statistical techniques for fraud detection, prevention, and assessment," *Mining massive data sets for security*, pp. 257–270, 2008.

[15] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.

[16] J. Park and J. Giordano, "Role-based profile analysis for scalable and accurate insider-anomaly detection," in *Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International*, April 2006, pp. 7 pp.–470.

[17] S. Nellikar, D. M. Nicol, and J. J. Choi, "Role-based differentiation for insider detection algorithms," in *Proceedings of the 2010 ACM Workshop on Insider threats* (Insider Threats '10). New York, NY, USA: ACM, 2010, pp. 55–62. Available: http://doi.acm.org/10.1145/1866886.1866897

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witte, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, vol. 11, no. 1, 2009.

[19] S. Mathew, M. Petropoulos, H. Ngo, and S. Upadhyaya, "A data-centric approach to insider attack detection in database systems," in *Recent Advances in Intrusion Detection*. Springer, 2010, pp. 382–401.

[20] M. Riveiro, "The importance of visualization and interaction in the anomaly detection process," *Innovative Approaches of Data Visualization and Visual Analytics*, p. 133, 2013.

[21] Z. Ferdousi and A. Maeda, "Unsupervised outlier detection in time series data," in *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*. IEEE, 2006.

[22] M. Thomas, L. Metcalf, J. Spring, P. Krystosek, and K. Prevost, "Silk: A tool suite for unsampled network flow analysis at scale," in *2014 IEEE International Congress on Big Data (BigData Congress 2014)*. IEEE, 2014, pp. 184–191.

[23] B. J. Biddle, *Role theory: Expectations, identities, and behaviors*. Academic Press, 2013.

[24] L. Shore, J. Coyle-Shapiro, and L. Tetrick, *The Employee-Organization Relationship: Applications for the 21st Century* (Applied Psychology Series). Taylor & Francis, 2012. Available: https://books.google.com/books?id=zQbDAgAAQBAJ

[25] D. Ferraiolo and D. Kuhn, "Natl Institute of Standards and Tech., Dept. of Commerce, Maryland, Role-Based Access Control," in *Proceedings of 15th Natl Computer Security Conference*, 1992.

[26] J. S. Park and S. M. Ho, "Composite role-based monitoring (CRBM) for countering insider threats," in *Intelligence and Security Informatics*. Springer, 2004, pp. 201–213.

[27] E. Bertino, P. Bonatti, and E. Ferrari, "TRBAC: A temporal role-based access control model," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 191–233, 2001.

[28] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "A generalized temporal role-based access control model," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 1, pp. 4–23, 2005.

[29] R. Simon and M. Zurko, "Separation of duty in role-based environments," in *Computer Security Foundations Workshop, 1997. Proceedings., 10th*. IEEE, 1997, pp. 183–194.

[30] R. Adaikkalavan, "Generalization and enforcement of role-based access control using a novel event-based approach," Ph.D. dissertation, University of Texas at Arlington, 2006.

[31] N. Baracaldo and J. Joshi, "A trust-and-risk aware RBAC framework: tackling insider threat," in *Proceedings of the 17th ACM symposium on Access Control Models and Technologies* (SACMAT '12). New York, NY, USA: ACM, 2012, pp. 167–176. Available: http://doi.acm.org/10.1145/2295136.2295168

[32] V. Frias-Martinez, J. Sherrick, S. Stolfo, and A. Keromytis, "A network access control mechanism based on behavior profiles," in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. IEEE, 2009, pp. 3–12.

[33] D. E. Denning, "An intrusion-detection model," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, vol. 13, no. 2, pp. 222–232, 1987.

[34] N. Clarke, F. Li, and S. Furnell, "A novel privacy preserving user identification approach for network traffic," *Computers & Security*, 2017.

[35] M. Vinupaul, R. Bhattacharjee, R. Rajesh, and G. S. Kumar, "User characterization through network flow analysis," in *Data Science and Engineering (ICDSE), 2016 International Conference on*. IEEE, 2016, pp. 1–6.

[36] F. Giroire, J. Chandrashekar, G. Iannaccone, K. Papagiannaki, E. M. Schooler, and N. Taft, "The cubicle vs. the coffee shop: behavioral modes in enterprise end-users," in *Proceedings of the 9th international conference on Passive and active network measurement* (PAM'08). Berlin, Heidelberg: Springer-Verlag, 2008, pp. 202–211. Available: http://portal.acm.org/citation.cfm?id=1791949.1791977

[37] A. Kind, M. P. Stoecklin, and X. Dimitropoulos, "Histogram-based traffic anomaly detection," *Network and Service Management, IEEE Transactions on*, vol. 6, no. 2, pp. 110–121, 2009.

[38] J. McHugh, R. McLeod, and V. Nagaonkar, "Passive network forensics: behavioural classification of network hosts based on connection patterns," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 99–111, April 2008. Available: http://doi.acm.org/10.1145/1368506.1368520

[39] N. Melnikov and J. Schönwälder, "User identification based on the analysis of network flow patterns," 2009. Available: http://www.faculty.jacobs-university.de/jschoenwae/nds-2009/melnikov-report.pdf

[40] S. Coull, F. Monrose, and M. Bailey, "On measuring the similarity of network hosts: pitfalls, new metrics, and empirical analyses," *Proceedings of the 18th Annual Network & Distributed System Security Symposium*, Feb. 2011.

[41] I. Paschalidis and G. Smaragdakis, "Spatio-temporal network anomaly detection by assessing deviations of empirical measures," *Networking, IEEE/ACM Transactions on*, vol. 17, no. 3, pp. 685–697, 2009.

[42] Y. Song, S. J. Stolfo, and T. Jebara, "Markov models for network-behavior modeling and anonymization," *Columbia Univ., New York, NY, USA, Tech. Rep., CUCS-029–11*, 2011.

[43] M. Kumpošt and V. Matyáš, "User Profiling and Re-identification: Case of University-Wide Network Analysis," *Trust, Privacy and Security in Digital Business*, pp. 1–10, 2009.

[44] G. Tan, M. Poletto, J. Guttag, and F. Kaashoek, "Role classification of hosts within enterprise networks based on connection patterns," in *Proceedings of USENIX Annual Technical Conference*, 2003, pp. 15–28.

[45] K. Xu, F. Wang, and L. Gu, "Network-aware behavior clustering of Internet end hosts," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 2078–2086.

[46] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia, "Reviewing traffic classification," in *Data Traffic Monitoring and Analysis*. Springer, 2013, pp. 123–147.

[47] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos, "Profiling the end host," *Passive and Active Network Measurement*, pp. 186–196, 2007.

[48] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. IEEE, 2009, pp. 268–273.

[49] A. Karasaridis, B. Rexroad, D. A. Hoeflin *et al.*, "Wide-scale botnet detection and characterization." *HotBots*, vol. 7, pp. 7–7, 2007.

[50] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 129–138.

[51] A. Vázquez, J. G. Oliveira, Z. Dezsö, K.-I. Goh, I. Kondor, and A.-L. Barabási, "Modeling bursts and heavy tails in human dynamics," *Physical Review E*, vol. 73, no. 3, p. 036127, 2006.

[52] Lucas, M., *Network Flow Analysis*. No Starch Press, 2010.

[53] Cisco Systems, Inc., "Cisco NetFlow Generation 3000 Series Appliances," [Online]. Available: http://www.cisco.com/go/nga., [Accessed: July 23, 2016].

[54] CERT/NetSA at Carnegie Mellon University, "SiLK (System for Internet-Level Knowledge)," [Online]. Available: http://tools.netsa.cert.org/silk., [Accessed: July 13, 2011].

[55] Cisco, "Introduction to Cisco IOS NetFlow - A Technical Overview," [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html., [Accessed: May 29, 2013].

[56] "SiLK FAQ," http://tools.netsa.cert.org/silk/faq.html, Carnegie Mellon University, accessed: June 1, 2015.

[57] B. Li, J. Springer, G. Bebis, and M. H. Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, no. 0, pp. –, 2013. Available: http://www.sciencedirect.com/science/article/pii/S1084804512002676

[58] Impulse Point, "SafeConnect: Network Access Control," [Online]. Available: http://www.impulse.com/., [Accessed: July 23, 2016].

[59] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Using low-rate flow periodicities for anomaly detection: Extended isi-tr-661," University of Southern California, Tech. Rep., 2009.

[60] J. Knockel and J. R. Crandall, "Protecting free and open communications on the internet against man-in-the-middle attacks on third-party software: We're foci'd." in *FOCI*, 2012.

[61] J. W. Tukey, "Box-and-whisker plots," *Exploratory Data Analysis*, pp. 39–43, 1977.

[62] E. Alpaydin, *Introduction to Machine Learning*. The MIT Press, 2010.

[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[64] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012. Available: http://doi.acm.org/10.1145/2347736.2347755

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California